[Microcontrollers & Embedded • Analogue • Audio • Digital • Test & Measurement]

September 2012

AUS$ 14.90 - NZ$ 17.90 - SAR 105.95 - NOK 102 £ 4.90

elektor

EMBEDDED LINUX

Software Development

+ Arduino on Course

1-bit Sound Generation ... What?!?

Nunchuk USB Interface

repurposing a video game controller

+ Model Train Interface

with USB connection and Windows script editor

+ PICo PROto

Minimalist prototyping tool for PIC16 or 18

09
9 770268 451173

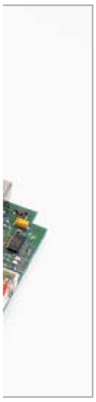
Do try this at home

I believe one our first articles linking DIY electronics to 'kiddie' electronics was the Gameboy Digital Oscilloscope (GBDSO) from 2000. Basically, you plugged a module into a Nintendo Gameboy game console and hey presto there's a simple portable oscilloscope. Initially we staged a Dutch-ish auction on the web to 'test the market', that is, check what price our customers were willing to pay for the product. Eventually the Elektor GBDSO reached sales volumes in the thousands across a period of more than ten years. If that GBDSO were an Olympic athlete, it would be in the company of Crescendo, EEDTs, Junior Computer, Filmnet Decoder, ATM18, SDR and Pico-C.
Resonating widely across the e-community, it is not surprising to see a product like GBDSO attracting strange questions from law-abiding readers (mostly from Germany), like "Where do I get hold of a known-good Gameboy, with full warranty, mint condition, at the lowest price?" to which we replied "Promise your kid(s) the new Advance model", or "Dig around in the sandpit at the local playground". Likewise, *Q*: "Did you get Nintendo's lawyers to approve that module?" *A*: "Hardly. Nintendo's techies subscribe to Elektor."
Twelve years on, I would unhesitatingly recommend to all you electronics designers out there to hack, fry, disembowel, blend, explore, rebuild or repurpose the tons of kid's electronics out there. It's cheap, often free and in plentiful supply. By rescuing stuff from skips and dumpsters filled by the throwaway generation, the electronics inside provides a mental link to the clever people who designed and build it all. Reverse engineering is good engineering. Do it creatively and with respect. Like we did with the Nunchuk on page 18.

Happy reading and nunchuking,
Jan Buiting, Managing Editor

CONTENTS

Volume 38
September 2012
no. 429



The Team

The Network

Tech the Future explores the solutions for a sustainable future provided by technology, creativity and science.

Our international teams

United Kingdom
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

USA
Hugo Vanhaecke
+1 860-875-2199
h.vanhaecke@elektor.com

Germany
Ferdinand te Walvaart
+31 46 4389417
f.tewalvaart@elektor.de

France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

Netherlands
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl

Spain
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

Brazil
João Martins
+55 11 4195 0363
joao.martins@editorialbolina.com

Portugal
João Martins
+351 21413-1600
joao.martins@editorialbolina.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+27 78 2330 694 / +31 6 109 31 926
j.dijk @elektor.com

China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Volume 38, Number 429, September 2012 ISSN 1757-0875

Publishers: **Elektor International Media,**
78 York Street, London W1H 1DP, United Kingdom.
Tel. +31 46 43 89 497 , fax: +31 46 43 70 161
www.elektor.com

The magazine is available from newsagents, bookshops and electronics retail outlets, or on subscription.

Elektor is published 11 times a year with a double issue for July & August.

Subscriptions: **Elektor International Media,**
78 York Street, London W1H 1DP, United Kingdom.
Tel. +31 46 43 89 497 , fax: +31 46 43 70 161
Internet: www.elektor.com/subs
Email: subscriptions@elektor.com

Rates and terms are given on the Subscription Order Form.

Head Office: **Elektor International Media b.v.**
P.O. Box 11 NL-6114-ZG Susteren The Netherlands.
Telephone: (+31) 46 4389444, Fax: (+31) 46 4370161

Distribution:
Seymour, 2 East Poultry Street, London EC1A, England.
Telephone:+44 207 429 4073

UK Advertising: **Elektor International Media b.v.**
P.O. Box 11 NL-6114-ZG Susteren The Netherlands.

Supporting Companies

Not a supporting company yet?

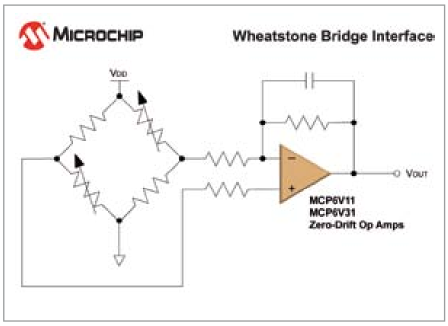
Contact Johan Dijk (j.dijk@elektor.com, +31 6 109 31 926)
to reserve your own space for the next edition of our members' magazine

Telephone: +31 46 4389444, Fax: +31 46 4370161
Email: j.dijk@elektor.com
Internet: www.elektor.com
Advertising rates and terms available on request.

Zero-drift opamps

for signal conditioning, instrumentation and portable sensor applications

Microchip Technology Inc., has broadened its portfolio of zero-drift operational amplifiers (op amps) with the debut of the MCP6V11 and MCP6V31 single amplifiers. Operating with a single supply voltage as low as 1.6 V and a quiescent current as low as 7.5 µA, these ultra-high-performance devices offer some of the industry's lowest quiescent current for the given bandwidth without sacrificing the optimal performance essential for portable applications in the



consumer, industrial and medical markets. With an aging world population in need of new therapies and early diagnostic tools, devices like the MCP6V11/31 enable the development of portable medical products integrated with higher efficiency, and signal conditioning hardware and software, which is critical to accommodate the continued push for lower costs and faster times to market. As well, designers of industrial applications — such as portable sensor conditioning and instrumentation requiring low power, smaller form factors, temperature considerations and cost-management — can benefit from the optimized performance, low quiescent current and low operating voltage made possible by the MCP6V11/31 op amps.

Employing Microchip's advanced CMOS technology, the devices require less current to operate the amplifier while simultaneously delivering longer battery life and minimal thermal-related challenges.

The self-correcting architecture of the MCP6V11/31 family provides a maximum input offset voltage of 8 µV for ultra-low-offset and low-offset drift, enabling maximum accuracy across time and temperature. The MCP6V11 offers 80 kHz of gain bandwidth product, with a low typical quiescent current of only 7.5 µA; while the MCP6V31 provides 300 kHz of gain bandwidth product, coupled with a low typical quiescent current of 23 µA.

Additionally, the MCP6V11 and MCP6V31 single amplifiers are both available in small 5-pin SOT-23 and 5-pin SC-70 packages, enabling minimal use of board space, ease of system design and reduced cost.

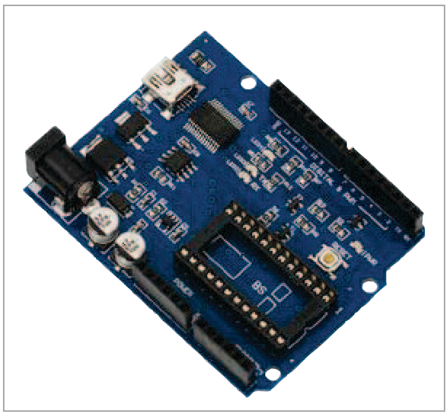
http://www.microchip.com/get/T15S

(120487-I)

Enter: Stampduino

Parallax' Stampduino is a BASIC Stamp development board designed to be compatible with most Arduino shields, reducing space and size requirements, with the purpose of optimizing the users systems.

All of Stampduino's 16 digital I/O pins are free to use, allowing you to fully utilize the capabilities of the BASIC Stamp. The board is compatible with most Arduino shields, for ease of use between systems. It has an integrated serial communication LEDs for a visual confirmation of data transfer, and a surface mount 3.3 V regulator to accommodate the incorporation of 3.3 V devices into your application. Stampduino is USB or externally powered, for those who



do rapid prototyping.

On the Parallax website, search part # 27140. The new product is priced at $29.99.

www.parallax.com

(120487-II

RFID SMT antenna

with comprehensive protection for automotive applications

PREMO launches a new family of its TP0702 standard, universally adopted by the industry. This format provides up to 50 mV/App/m (for 7.2 mH) sensitivity which gives it the best sensitivity in this transponder size. The new TP0702U and TP0702UCAP is an SMD antenna for low frequency 20 kHz-150 kHz receiver applications. TP0702UCAP provide an upper and lateral side protection with co-polyamide polyhexamethylene polymer walls, gamma radiated with high thermal stability (supports up to 290 º C) and mechanical resistance (exceeds 150 Mpa of mechanical strength).

This antenna features a NiZn ferrite core with high surface resistivity (>10 MΩ/mm) that provides a highly stable behavior (better than ±3%) over a wide temperature range (–40 º C to 125 º C).

The new TP0702UCAP, is particularly suitable for applications such as TPMS (Tire Pressure Monitoring Systems) which requires an excellent performance under extreme conditions, according to AEC-Q200 and additional requirements as EU regulations.

PREMO offers four standard values, 2.38 mH, 4.91 mH, 7.2 mH and 9 mH at 125 kHz. Other inductance values and frequencies, from 340 µH to 18.5mH, are available upon request.

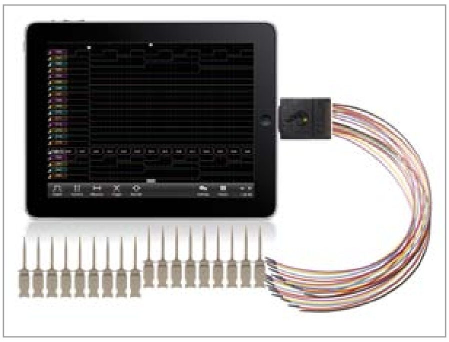
A surface mount (SMT) device, the new antenna allows easy use in the automated process of mounting circuit boards, thus eliminating any manual handling.

www.grupopremo.com

(120487-IV)

iPad transformed into portable logic analyzer

Oscium's new LogiScope is a logic analyzer with the real time data analysis capabilities of an oscilloscope. Oscium's test and measurement equipment is ultra-portable and designed specifically for the iOS family of products like the iPhone, iPad and iPod touch.



The LogiScope is a powerful tool that transforms an iPad into a 100 MHz, 16 channel logic analyzer for only $389.99. Traditionally, a logic analyzer records a buffer which has to be downloaded and searched. Now with LogiScope's advanced triggering, decoded data can be viewed live, eliminating the need to capture, pause, and then view. There is no need to settle for pictures when it's possible to analyze live data.

The touchscreen-based iOS platform is truly a superior solution making the display simple and intuitive. For example, changing the timescale is as easy as zooming into a picture on a smartphone, and adjusting the delay is as simple as a swipe across the top of the screen. LogiScope's intuitive interface also provides immediate feedback for signals that are too fast for the timescale by changing the waveform to red. No need to wait for the reading to be complete. Save precious time and receive instant feedback with LogiScope.

"The user interface on LogiScope is extremely well executed," said Bryan Lee, President of Oscium. *"It's our best interface yet."*

LogiScope version 1.0.12 is available for download free in the Apple App Store. The LogiScope app is made for: iPod touch (3rd, and 4th generation), iPhone 4S, iPhone 4, iPhone 3GS, iPad 3, iPad 2, and iPad.

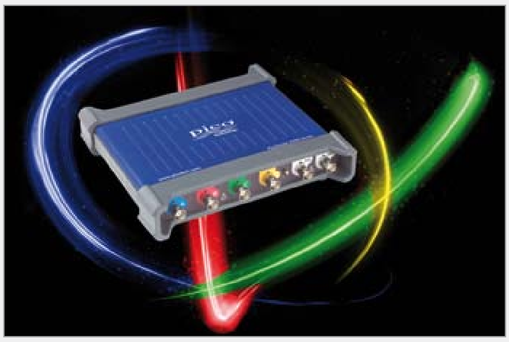
LogiScope hardware can be purchased for $389.99 from Oscium directly or from one of their partners.

www.oscium.com (120487-III)

New 4-channel, compact, USB-powered oscilloscopes

The PicoScope 3000 Series of high-performance oscilloscopes has been expanded to include six new 4-channel models. The new oscilloscopes offer a maximum sampling rate of 1 GS/s (up to 10 GS/s effective for repetitive signals), a range of input bandwidths from 60 MHz to 200 MHz, and buffer memory depths from 4 M to 128 M samples. The new FlexiPower™ system allows the scopes to run on either USB or AC power. With an option of either a built-in function generator or a built-in arbitrary waveform generator, and a new, slim case design, these scopes are perfect for engineers and technicians needing a complete, portable test bench in a single unit.

The PicoScope oscilloscope software includes as standard all the oscilloscope and spectrum analyzer functions you would expect, as well as serial decoding, mask limit testing, segmented memory and advanced triggers: features that often cost extra on other manufacturers' scopes. Running on your Windows PC, PicoScope shows waveforms on a large, clear display and allows easy zooming and panning under keyboard or mouse control. Other built-in features include persistence displays with fast waveform update rates, math channels, automatic measurements with statistics, programmable alarms, and decoding of I^2C, UART/RS232, SPI, CAN bus, LIN and FlexRay signals. Updates to the software are released regularly, free of charge.



The advanced triggering modes include pulse width, interval, window, window pulse width, level dropout, window dropout, runt pulse, variable hysteresis, and logic. All triggering is digital, ensuring lower jitter, greater accuracy and higher voltage resolution than the analog triggering found on many competing scopes.

A free Software Development Kit (SDK) allows you to control the new scopes from your own custom applications. The SDK includes example programs in C, C++, Excel and LabVIEW, and can be used with any language that supports C calling conventions. The PicoScope software and SDK are compatible with Microsoft Windows XP, Windows Vista and Windows 7.

The PicoScope 3000 Series 4-channel oscilloscopes are available now from Pico distributors worldwide and from www.picotech.com. Prices range from only £599 for the 60 MHz PicoScope 3404A with function generator to only £1349 for the 200 MHz PicoScope 3406B with AWG, including four probes and a 5-year warranty.

www.picotech.com (120487-VIII)

Cloud-ready HMI extension

for industrial panel PC family

Input device specialist Hoffmann + Krippner has further extended its flexx-IPC™ customizable panel PC concept with the launch of a software platform that will reduce the time and cost of human machine interface (HMI) development and deployment. Using flexx-HMI OEMs can rapidly create optimized HMIs that seamlessly integrate with industrial automation, process control and other deterministic PLC- and controller-based applications.

The new flexx-HMI® technology allows OEMs to quickly and easily develop advanced and intuitive HMIs that harness the power of Hoffman + Krippner's Microsoft Windows Compact 7-based flexx-IPC panel PCs. These PCs use the company's innovative membrane input technology to support cost-effective front panel customization in order volumes as low as just 20 units. A selection of flexx-HMI drivers supports more than 100 PLCs and controllers, while the HMI software provides comprehensive 'cloud-based' support for secure, remote access via any web-enabled device.

OEMs looking to create advanced user interfaces for the platform can purchase a comprehensive development environment from Hoffmann & Krippner. Using project wizards this development environment dramatically simplifies the building of HMI applications including data source creation, screen design, testing and deployment. Once the HMI has been created it is possible to provide users with access to a cloud-based service that supports monitoring and control access from any browser on any hardware.

www.flexx-ipc.co.uk (120487-VI)

UV cure adhesive

for optoelectronic and circuit assembly applications

Engineered Material Systems, introduces 535-10M-1 UV Cured Adhesive formulated for disk drives, camera modules, optoelectronic and circuit assembly applications. 535-10M-1 is an ultra low

Low Cost. High Performance.

PC-based measurements from only £69

Multifunction DAQ

NI USB-6008

12-Bit, 10 kS/s, Multifunction DAQ

- 8 analogue inputs; 2 analogue outputs; 12 digital I/O; 32-bit counter
- Bus-powered for high mobility; built-in signal connectivity
- NI-DAQmx driver software and LabVIEW SignalExpress LE interactive data-logging software
- *£99*

ALSO AVAILABLE:

NI USB-6009 14-Bit, 48 kS/s, Multifunction DAQ - *£179*
NI USB-6210 16-Bit, 250 kS/s, Multifunction DAQ - *£399*
NI USB-9201 12-Bit, 8 Channels, 500 kS/s - *£549*

Sensor-based measurements

NI USB-TC01

Thermocouple Measurement Device - *£79*

ALSO AVAILABLE:

NI USB-9211A 24-Bit, 4 Channels, Thermocouple Input - *£529*
NI USB-9219 4 Channels, Universal Analogue Input - *£1049*

Digital I/O

NI USB-6501

24 Channel, 8.5 mA, USB Digital I/O - *£69*

ALSO AVAILABLE:

NI USB-6525 8 Solid-State Relays, 8 DI, Counter, Channel-to-Channel Isolated - *£249*
NI USB-9421 8 Channels, 11-30V Digital Input - *£339*
NI USB-9472 8 Channels, 6-30V Digital Output - *£339*

Instruments

NI USB-5132

50 MS/s Bus-Powered Digitiser/Oscilloscope - *£499*

ALSO AVAILABLE:

NI USB-5133 100 MS/s Bus-Powered Digitiser/Oscilloscope - *£699*
NI USB-4065 6½-Digit USB Digital Multimeter - *£1049*

Instrument Control

NI GPIB-USB-HS

GPIB Controller for Hi-Speed USB 2.0 - *£349*

ALSO AVAILABLE:

NI USB-232 Single-Port RS232 Interface for USB - *£129*
NI USB-485 Single-Port RS485 Interface for USB - *£129*
NI USB-8451 I2C/SPI Interface - *£339*

At lower cost than you might think, National Instruments provides an intuitive platform that can connect to thousands of sensors and instruments to acquire, analyse and process signals, and present results graphically. Whether programming in C, Basic or LabVIEW on Windows, Mac or Linux, get the latest in PC-based data acquisition, instrumentation and instrument control, starting at **less than £70**.

>> Buy online and explore the range at **uk.ni.com/lowcost**

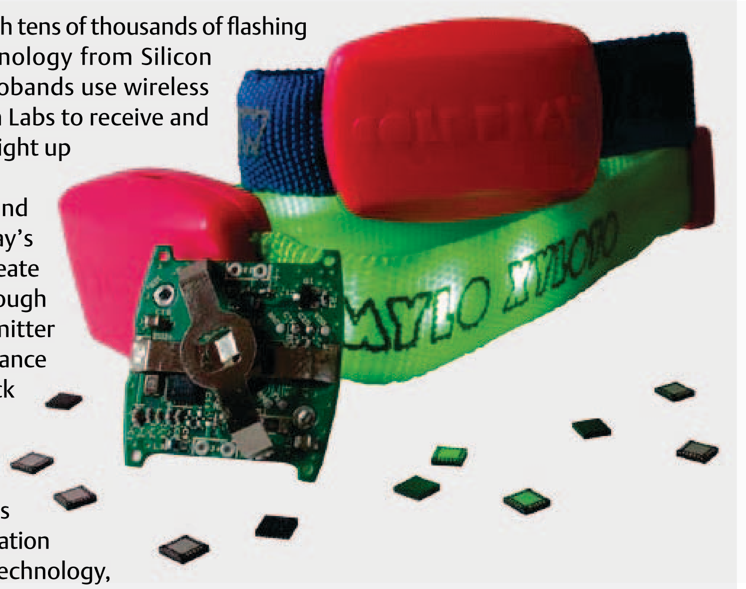
01635 517300
uk.ni.com
info.uk@ni.com

Join us to improve your product skills, knowledge and productivity using NI LabVIEW software, 9 UK & Ireland locations in September and October: **uk.ni.com/devdays**

JOIN THE CONVERSATION: *Follow us on* *Search* **niukie**

NATIONAL INSTRUMENTS

Xylobands™ LED wristbands get crowds rocking at Coldplay concerts



Nothing lights up a rock arena quite like a Coldplay audience with tens of thousands of flashing Xylobands(TM) LED wristbands powered by embedded technology from Silicon Laboratories Inc. Created by UK-based RB Concepts Ltd., Xylobands use wireless ICs and ultra-low-power microcontrollers (MCUs) from Silicon Labs to receive and process wireless signals that trigger each wristband's LEDs to light up in sync with the music and stage lightshow.
The Xylobands are the unique, patented creation of inventor and Coldplay fan Jason Regler, a co-owner of RB Concepts. Coldplay's high-energy music and lyrics inspired Regler's bright idea to create a wireless LED wristband that could be controlled remotely through proprietary software and a laptop connected to a radio transmitter to enable fans to be part of the lightshow. Recognizing the brilliance of Regler's invention, the Brit Awards- and Grammy-winning rock band has used Xylobands to light up arenas and stadiums all around the world.
"Taking the LED wristband from concept to finished product required best-in-class embedded control and wireless technology," said Jason Regler, director of technology and innovation at RB Concepts. "Silicon Labs was the ideal choice for wireless technology, enabling us to achieve both FCC and Industry Canada Certification and deliver more than 30,000 Xylobands just in time for a recent Coldplay concert in Edmonton, Alberta, Canada."
"RB Concepts and Silicon Labs have been outstanding," added Phil Harvey, Coldplay's creative director. "They've delivered hundreds of thousands of units whenever and wherever we've needed them. The wireless LED wristbands have broken down that invisible wall between band and audience and put the audience right at the heart of the show. The mass feeling of joy and wonder when they all light up at the top of the show is hard to put into words."
Xylobands have a very broad appeal. In addition to lighting up rock concerts, Xylobands can generate interactive audience participation at a wide variety of sporting events, theme parks, festivals, parties and corporate activities.
"Xylobands and the low-power, long-range wireless technology behind the product is a game changer for how audiences can interact with performers and become an integral part of the concert experience," said Diwakar Vishakhadatta, vice president and general manager of Wireless Embedded Systems at Silicon Labs. "RB Concepts' wireless wristbands are also a versatile innovation that can be applied to a wide range of events and activities."

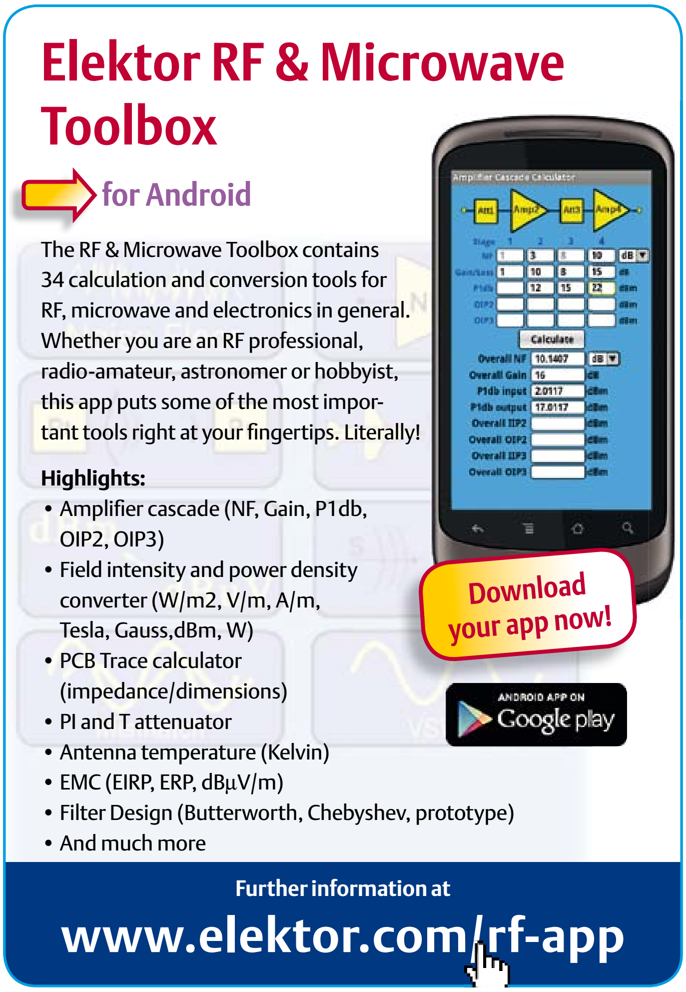
Seeing is believing. Visit www.coldplay.com to see Xylobands flashing on and off to Coldplay's hit song, "Charlie Brown," performed at Rexall Place in Edmonton.
Silicon Labs' EZRadioPRO transmitters enable Xylobands base stations to transmit wireless signals in the sub-GHz frequency bands. These transmitters offer industry-leading RF performance resulting in exceptional wireless range and compliance with stringent wireless regulatory standards. Xylobands LED wristbands include EZRadio receiver ICs designed for low-power sub-GHz radio applications. These receivers offload many RF-related activities from the system MCU, allowing extended MCU sleep periods and resulting in lower power consumption. The EZRadio products work in concert with Silicon Labs' ultra-low-power C8051F98x MCUs, offering the industry's lowest active mode current consumption, which saves power when the application is running, as well as the industry's lowest current consumption in sleep mode, making it an ideal choice for battery-powered wireless applications.

www.silabs.com (120487-VII

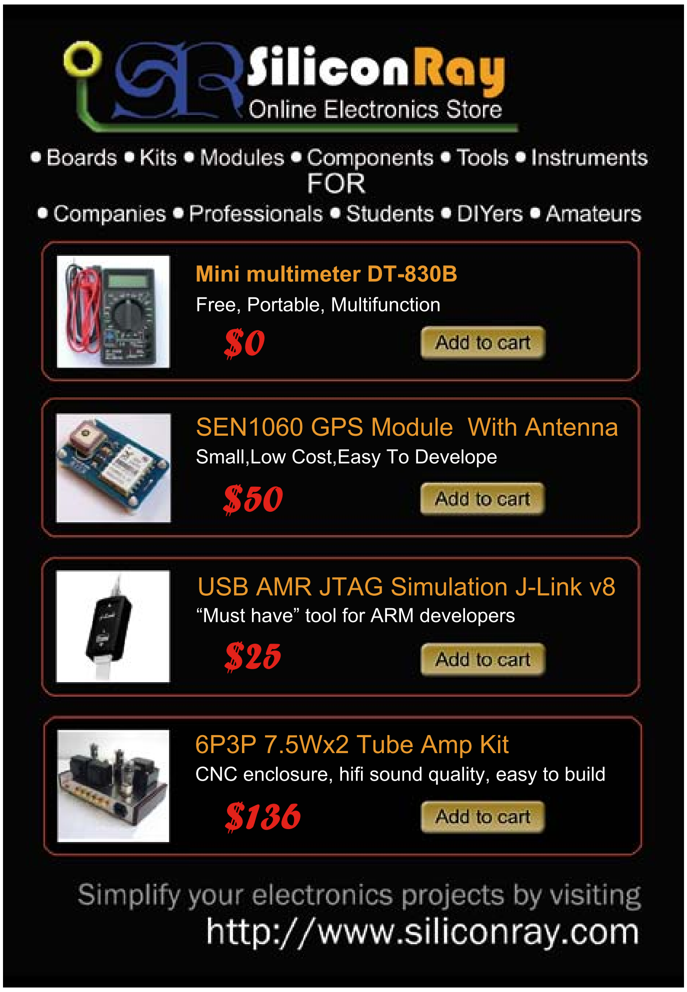
stress, lower glass transition temperature version of the 535-10M UV Cure Adhesive. The material is designed to eliminate any "crowning" (warpage) of sliders in head gimbal assemblies and can be used in other bonding applications in the head stack assembly. The material also can be used for lens bonding in camera modules, chip encapsulation in smart cards and a variety of general bonding applications in photonics assembly.
This new nonconductive UV cured adhesive

cures rapidly when exposed to high-intensity UV light. 535-10M-1 is a low outgassing, extremely flexible, high-strength epoxy adhesive that does not contain antimony.
The 535-10M-1 was developed to pass the rigorous reliability requirements in disk drive, camera module, photonics and circuit assembly applications. 535-10M-1 is the latest addition to Engineered Material Systems extensive line of electronic materials.

www.conductives.com (120487-V)



Elektor RF & Microwave Toolbox
for Android
The RF & Microwave Toolbox contains 34 calculation and conversion tools for RF, microwave and electronics in general. Whether you are an RF professional, radio-amateur, astronomer or hobbyist, this app puts some of the most important tools right at your fingertips. Literally!
Highlights:
• Amplifier cascade (NF, Gain, P1db, OIP2, OIP3)
• Field intensity and power density converter (W/m2, V/m, A/m, Tesla, Gauss,dBm, W)
• PCB Trace calculator (impedance/dimensions)
• PI and T attenuator
• Antenna temperature (Kelvin)
• EMC (EIRP, ERP, dBμV/m)
• Filter Design (Butterworth, Chebyshev, prototype)
• And much more
Download your app now!
ANDROID APP ON Google play
Further information at
www.elektor.com/rf-app



SiliconRay
Online Electronics Store
• Boards • Kits • Modules • Components • Tools • Instruments
FOR
• Companies • Professionals • Students • DIYers • Amateurs
Mini multimeter DT-830B
Free, Portable, Multifunction
$0
Add to cart
SEN1060 GPS Module With Antenna
Small,Low Cost,Easy To Develope
$50
Add to cart
USB AMR JTAG Simulation J-Link v8
“Must have” tool for ARM developers
$25
Add to cart
6P3P 7.5Wx2 Tube Amp Kit
CNC enclosure, hifi sound quality, easy to build
$136
Add to cart
Simplify your electronics projects by visiting
http://www.siliconray.com

THE ORIGINAL SINCE 1994
PCB-POOL®
Beta LAYOUT
PCB Prototypes and small series
EAGLE order button
pcb-pool.com/download-button
20% off on your first PCB order
FREE Stencil
with every PCB Prototype order
FREE Stencil!
Discount
Evaluate 5 orders and get 10% off
Free Phone UK: 0800 389 8560
sales@pcb-pool.com
www.pcb-pool.com
Beta LAYOUT
create : electronics

eSTORE®
Beta LAYOUT
Develop, assemble, solder
€ 36,50*
Arduino Mega (ATMega 1280-16AU) compatible
Reflow Controller
€ 129,00*
UHF RFID Starterkit Basic
€ 279,00
Large Beta-Reflow-Kit
€ 129,00*
* Incl. VAT and exclusive of shipping costs
www.beta-eSTORE.com
Beta LAYOUT
create : electronics

DesignSpark
chipKIT™
Challenge

DesignSpark

Beginning in September 2011, design engineers from around the globe were challenged to turn their hot ideas into cool solutions using Microchip's chipKIT Max32 development board. While these energy-conscience designers connected in the DesignSpark forums, their projects continued to evolve until the final innovative and eco-friendly project designs were ready for the judges.

The judges reviewed all the entries and scored the projects based on technical merit, originality, design optimization, and quality of an extension card using DesignSpark's PCB tools. The judge's results are now final. Thanks to everyone who participated in this incredible competition and congratulations to all the winners! Let your international recognition and prestige begin.

First Prize

Dean Boman

United States – d.boman@cox.net

Energy Monitoring System

The Energy Monitoring System (EMS) provides real-time home electrical usage data to occupants so they can make informed consumption-related decisions. The innovative system features a chipKIT Max32 development board along with two extension boards. A web server provides usage tracking on a circuit-by-circuit basis. It interfaces with a home automation system for long-term monitoring and data logging. The system utilizes custom software written in C using Microchip's MPLAB development environment and is integrated with the Microchip TCPIP stack.

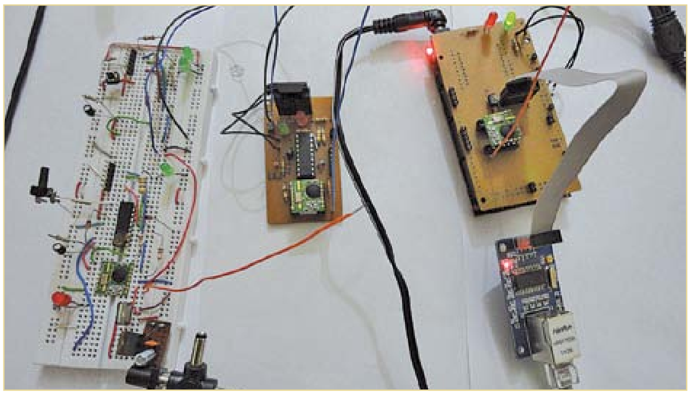
Second Prize

Raul Alvarez Torrico

Bolivia – raul-at@hotmail.com

Home Energy Gateway

The well-designed Home Energy Gateway enables users to monitor energy consumption and control household devices (e.g., lights) remotely. A chipKIT Max32-based embedded Gateway/Web Server communicates with two kinds of smart devices in a house: one smart meter that monitors average active real power consumption and several Smart Plugs in a Home Area Wireless Network. A user can monitor the Smart Plugs and adjust them with a web interface.



ChipKIT™ Challenge Winners

Third Prize

Graig Pearen

Canada – graig@pearen.ca

SunSeeker (PV Array Tracker)

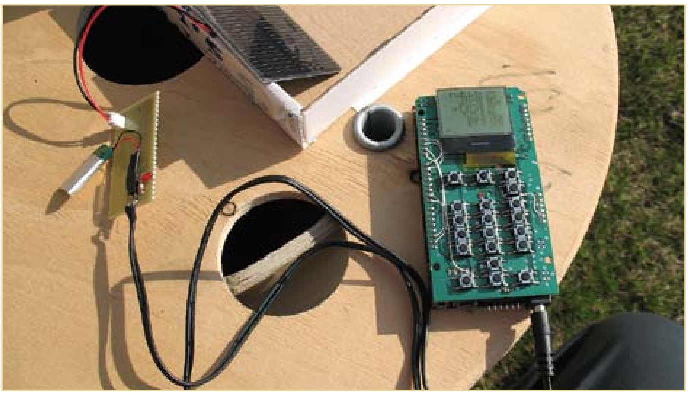
Imagine having the ability to control solar photovoltaic (PV) arrays to continually receive the most sunshine for optimal energy conversion. The SunSeeker enables you to do just that. Featuring a Microchip Technology chipKIT Max32, the system is designed to track, monitor, and adjust PV arrays based on weather and sky conditions. It identifies conditions, measures PV and air temperature, compiles statistics, and communicates with a local server that enables the SunSeeker to facilitate software algorithm development and refinement. Diagnostic software monitors the design's motors to show both movement and position.

Honourable Mention

Jaromir Sukuba

Slovakia – j.sukuba@gmail.com

Handheld PIC18 IDE



The Handheld PIC18 IDE is an autonomous system for creating, editing, and assembling source files for a Microchip Technology PIC18. Binary output of this process is programmed into a target PIC18 or is debugged at source level, called PP4. The hardware is simple. It consists of a user interface (LCD and keyboard), data storage, and a programming interface. The handy IDE also includes a BF interpreter for writing and executing scripts in BF language. You can use BASIC with the tag-along BASIC interpreter. The device is solar powered with a Li-Ion cell backup.

Honourable Mention

John Schuch

United States – hackersbench@gmail.com

Wireless Mesh Network Time Server

The solar-powered Wireless Mesh Network Time Server receives time and date data from a GPS receiver module and rebroadcasts it into a mesh network through an XBee module for use by any devices attached to the network. An NiCd battery pack powers the server and a small PV panel recharges it. In Normal mode, the server broadcasts the time and date once per minute. and If another device on the network requests it, the server switches to Stream mode in which the transmission speed increases to once per second. The battery pack's voltage is measured and transmitted in Test mode. The relatively simple software enables you to program in a high-level language (MPIDE) while focusing on time and energy conservation.

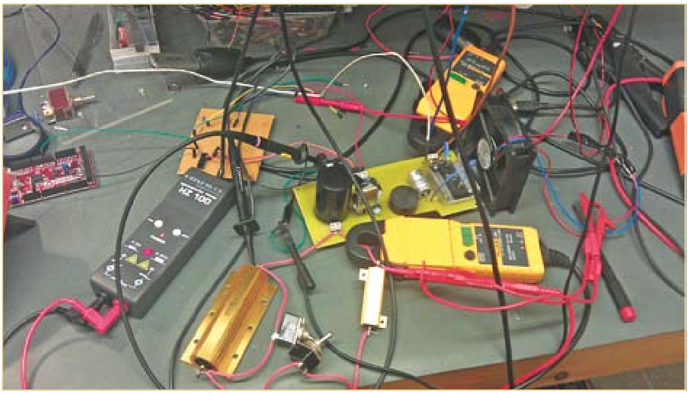
Honourable Mention

Ian Johnson, Sajjad Lalji, David Weight

United Kingdom – ian.johnson@wattcircuit.com

MPPT Boost Converter

Maximum power point trackers (MPPTs) are used to ensure that maximum power is transferred to a device from a range of renew-



able energy input applications, such as thermoelectric generators (TEGs), photovoltaic (PV) panels, and inductive power transfer (IPT) systems. An impedance-matching method was developed for a closed-loop digital control system. It provides MPPT for input applications that have an approximately fixed internal resistance, such as TEGs and IPTs. The method has been analysed and used to construct the prototype converter with an MCU performing control operations. A modified model might be suitable for input applications that have a variable internal resistance due to temperature, light or both.

Honourable Mention

Manuel Iglesias Abbatemarco

Venezuala – mhanuel@ieee.org

Eco-Friendly Home Automation Controller

Built around a chipKIT Arduino-compatible board, this well-planned system connects to a custom chipSolar board of the same footprint that provides power from two Li-Ion cells. The board implements an MPPT charger that deals with a solar panel's nonlinear output efficiency. A custom chipWireless board comprising a Quad Band

GSM/GPRS modem, an XBee socket, an SD card connector, and companion RTCC provides connectivity. The software was written using MPIDE. The SD card is used to log data from sensors and future memory requirements.

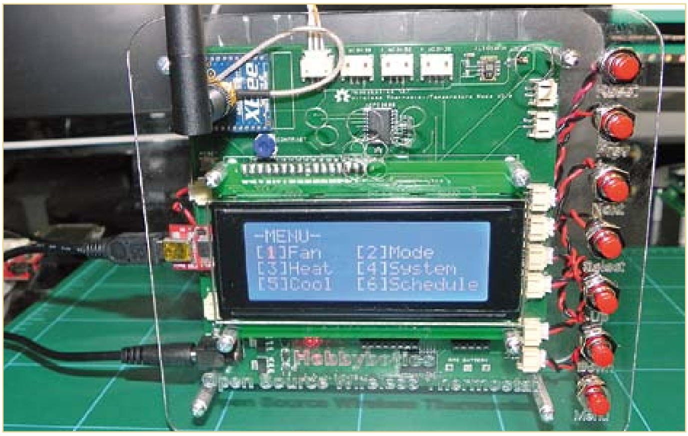
Honourable Mention

Curtis Brooks

United States – brooksware2000@gmail.com

Internet-Enabled Multizone Thermostat

An Internet-enabled thermostat gives users maximum control of building temperatures. This novel system features three main sections: an XBee shield, a wireless temperature board, and an I²C-controlled output board.



It includes two wireless temperature control boards and two I²C output boards. A router provides Internet access. The first node comprises a chipKIT Max32, a Max32 Ethernet Shield, and an XBee Shield (PCB). The second node is the wireless thermostat that controls an HVAC system and receives data from room nodes. The room nodes control a motorised damper to regulate each room's temperature.

(120420)

The complete DesignSpark/ChipKITTM Design Challenge file repository is accessible at

www.circuitcellar.com/contests/chipkit2012

Nunchuk USB interface

repurposing a video game controller

By Anthony Le Cren (France)

The Wii games console is supplied with an accessory called Nunchuk: a second controller that has a 3-axis accelerometer, an analogue joystick, and two buttons. All it takes is a PIC18F2550 to communicate with this man/machine interface using the I²C protocol and use it for other applications, e.g. in robotics, modelling, for DMX, etc.

The famous Wii games console from Japanese multinational Nintendo uses a Bluetooth wireless controller called a Wiimote. This can be connected by cable to another control unit called Nunchuk, which enables the player to use both hands in a video game, Wiimote in one hand and Nunchuk in the other. What interests me here is to re-purpose the Nunchuk controller with the help of a board based on the PIC18F2550 µC. This pretends to be the Wiimote, and in this way accesses the data contained in the Nunchuk using the I²C communication protocol. Once the data have been recovered, the user will have several options for using them:

- displaying them on an LCD screen;
- sending them to the PC via a man/machine interface (mouse, keyboard);
- using them to control actuators;
- redirecting them to an RS232 link so they can be used with another microcontroller.

These data comprise indications corresponding to the three X, Y, and Z axes of the accelerometer, the analogue joystick position, and the status of the two buttons called C and Z. We'll look into all this in more detail later. And there you have the principle of my Nunchuk re-purposing interface (**Figure 1**).

Operation

As we would expect, the whole thing revolves around the 18F2550 microcontroller, here associated with a 20 MHz crystal, whose frequency deserves a little explanation. This µC can be configured in 12 different ways. Now for the clock frequency, the use of the USB bus requires a multiple of 12 MHz. This is why the 20 MHz crystal frequency is divided by 5, before driving an internal PLL running at 96 MHz. This frequency is then finally divided by two, i.e. a final frequency of 48 MHz. This detail is not irrelevant, as it needs to be taken into account during program development. You can find the oscillator configuration block diagram in the PIC manufacturer's documentation (p.24 Oscillator types) [4].

Nunchuk interface characteristics

- controller connected directly to the main PCB
- I²C protocol
- allows reading of the digital data from accelerometer (10 bits), analogue joystick (8 bits), and buttons (active Low)
- USB connector for interface with PC
- PIC programming with a bootloader and PDFSUSB software
- application programming using Flowcode
- DB9 sockets compatible with E-blocks modules
- 4-pin expansion port for a future serial link

The 5 V supply is provided via the USB cable (K2) connected to the PC. The supply voltage for the controller is reduced to 3.3 V by the LP2950ACZ-3.3 regulator (IC2). The I²C bus pull-up resistors are included in the controller, so there's no point providing any on our board.

This interface board also fulfils the role of a development board. This is why all the microcontroller pins are made available on two DB9 connectors: K4 carries all the 18F2550's analogue input pins, K5 is wired to the six MSBs of Port B, whose port labels have been designated specially to take a standard liquid crystal display with two lines of 16 characters. But obviously this accessory is not obligatory. Port B's two LSBs are reserved for the I²C bus (K2) for communicating with the controller, while K3 has been provided for a future external UART serial link.

It's easy to add other extensions, detectors, or other peripherals. LEDs D1–D4, active high, are available for your own applications.

There are two buttons, one for initializing the microcontroller (S2) and the other for accessing the PIC's bootloader (S1).

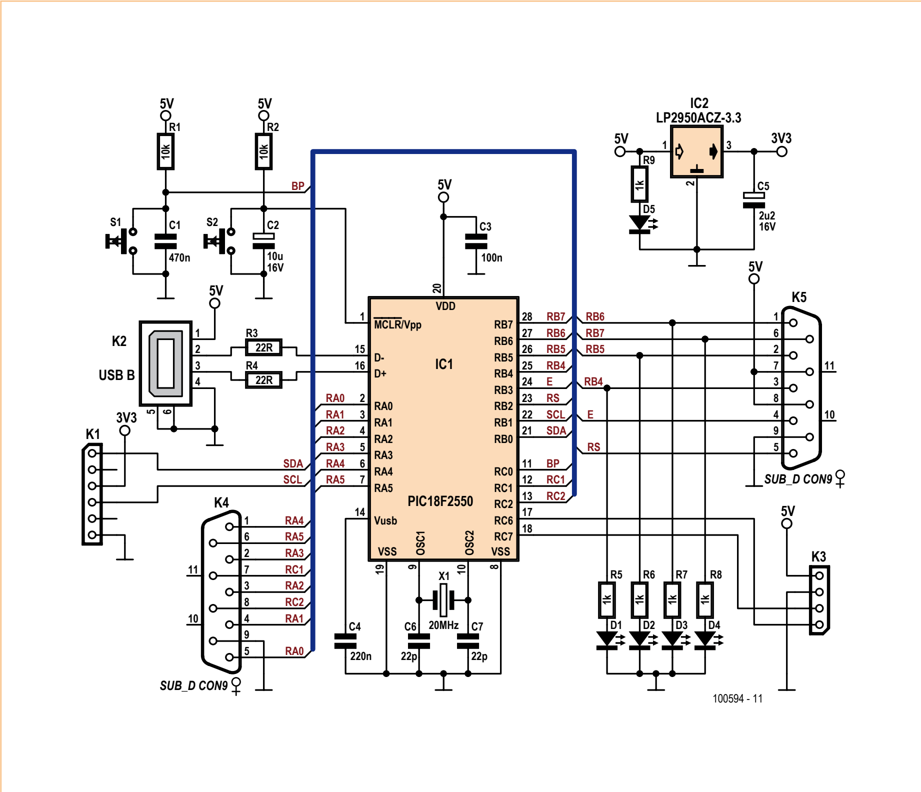


Figure 1. The circuit diagram of the interface board consists of little more than a PIC surrounded by a few connectors.

To measure the I²C frame (**Figure 2**) using an oscilloscope or some other bus analyser tool, all you have to do is pick up the signal on the SCL and SDA line, along with the RC2 line for synchronization (K4 pin 8) – and not forgetting the ground connection. This SYNCRO signal goes high when we start to read the data from the Nunchuk.

There are lots of possible applications for an accelerometer like the one in the controller. Hence it's important to be able to modify the examples provided on the Elektor website [1] to suit yourself, whence the use of a bootloader, as for many microcontroller applications.

Once the Nunchuk.hex firmware [1] has been loaded into the PIC, the user will be able to access the contents of the 32 kB Flash ROM

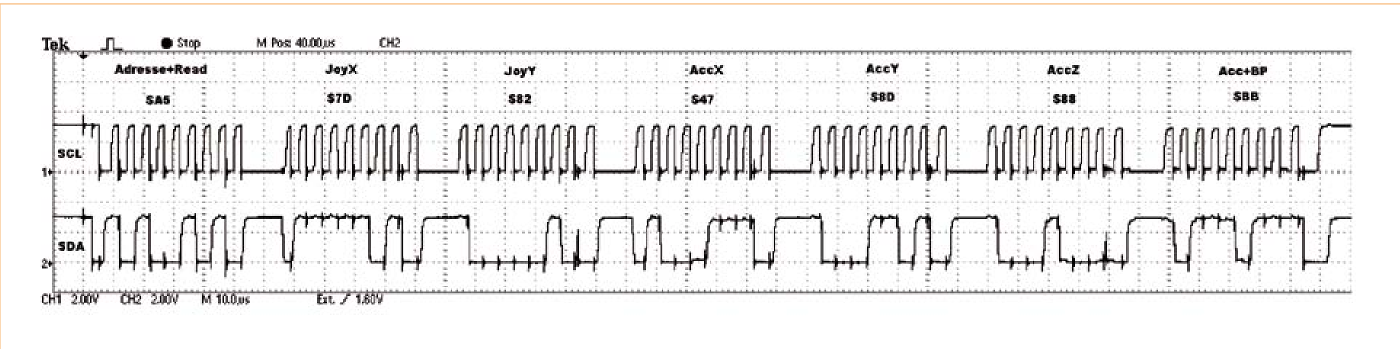


Figure 2. The whole frame is read in seven bytes.

Table 1.

Operation	address								R/W	hex
	D7	D6	D5	D4	D3	D2	D1	D0		
write: transfer of data from the interface to the Nunchuk controller	1	0	1	0	0	1	0	0		$A4
read: transfer of data from the Nunchuk controller to the interface	1	0	1	0	0	1	0	1		$A5

Table 2. Initialising the Nunchuk controller

start bus I²C	The I²C bus data flow from the interface board to the Nunchuk controller (Figure 3)
send 0xA4	
send 0xF0	
send 0x55	
stop bus I²C	
start bus I²C	
send 0xA4	
send 0xFB	
send 0x00	
stop bus I²C	

Table 3. Initialising the pointer in the Nunchuk controller RAM

start I²C bus	The I²C bus data flow from the interface board to the Nunchuk controller (Figure 4)
send 0xA4	
send 0x00	
stop I²C bus	

Table 4. Reading the data from the Nunchuk controller

start I²C bus	The I²C bus data flow from the Nunchuk controller to the interface board, apart from the 1st byte 0XA5 (read command) (Figure 4)
send 0xA5	
receive joystick X from **NUN_BUF[0]**	
receive joystick Y from **NUN_BUF[1]**	
receive accelerometer X (MSB) from **NUN_BUF[2]**	
receive accelerometer Y (MSB) from **NUN_BUF[3]**	
receive accelerometer Z (MSB) from **NUN_BUF[4]**	
receive accelerometer LSBs and button C & Z status from **NUN_BUF[5]**	
stop I²C bus	

using the Pdfsusb software supplied by Microchip [3] and in this way adapt the example programs to your requirements.

I²C protocol

On the I²C bus, the PIC will always be the master and the controller the slave. The Nunchuk's 7-bit I²C address is defined and set by the manufacturer as $52 (`1010010`). There are other peripherals, like the standard Wii controller, that can also be connected to the Wiimote. So each Wii peripheral has its own address.

As for any I²C bus, the first byte transmitted by the master is not data but an address. The format of the address byte is a bit special, as the function of the D0 bit (R/W) is to indicate if the master is requesting a read from the slave or conversely, if the master is forcing the slave to write. Starting from the 7-bit address code $52, this gives the two command bytes given in **Table 1**.

Before starting to read the data from the controller, we need to write initialize it in order to access the registers we're interested in, using the bit sequence given in **Table 2**.

You'll find more detailed explanations of these internal registers from the link [2].

Then comes a reset for the RAM pointer to the area containing the controller data by the sequence (**Figure 4**) in **Table 3**.

And to end, the whole frame (**Figure 2**) is read in seven bytes:

- start condition
- 1 address byte plus the read operation ($A5) sent by the PIC (master)
- 6 data bytes transmitted by the controller (slave)
- stop condition

Table 5. I²C protocol for reading the Nunchuk controller

byte 1 0xA5	byte 2 NUN_BUF[0]	byte 3 NUN_BUF[1]	byte 4 NUN_BUF[2]	byte 5 NUN_BUF[3]	byte 6 NUN_BUF[4]	byte 7 NUN_BUF[5]
address Nunchuk + read operation	joystick X (0–255) 8 bits	joystick Y (0–255) 8 bits	accelerometer X MSB (bits 9–2)	accelerometer Y MSB (bits 9–2)	accelerometer Z (bits 9–2)	accelerometer LSBs + buttons C & Z

Table 6.

D7	D6	D5	D4	D3	D2	D1	D0
accel. Z	accel. Z	accel. Y	accel. Y	accel. X	accel. X	button C	button Z
bit 1	bit 0	bit 1	bit 0	bit 1	bit 0	active low	active low

The six data bytes received will be stored in the form of a table of values in the 18F2550's RAM (**NUN_BUF[0]** to **NUN_BUF[5]**) as shown in **Tables 4 & 5**.

As the accelerometer word uses a 10-bit binary format, the LSBs are recovered in the last byte. We need to concatenate the word in order to obtain a value between 0 and 1023 (see the detail of this binary coding in **Table 6** below).

The values for the three axes are determined through the following calculation ('bouton' = button):

```
joy_x = NUN_BUF[0]
joy_x = NUN_BUF[1]
accel_x = (NUN_BUF[2]*4)+((NUN_BUF[5]/4) AND 0x03)
accel_y = (NUN_BUF[3]*4)+((NUN_BUF[5]/16) AND 0x03)
accel_z = (NUN_BUF[4]*4)+((NUN_BUF[5]/64) AND 0x03)
bouton_c = (NUN_BUF[5] /2 ) AND 0x01
bouton_z = NUN_BUF[5] AND 0x01
```

Construction

As there are no SMD devices, fitting the components doesn't present any special problems (**Figure 9**). It's best to start by soldering the smallest components, then the IC socket, the LEDs and the capacitors, and then last of all the crystal and the connectors. But before that, the connection from the Nunchuk to the PCB (K2) deserves a special mention: you need to make a double cut-out in the PCB using a mini disc-cutter so as to be able to plug the controller's original connection in. There's no actual connector on the PCB – the contact is made directly with the copper tracks. Take care to insert it the right way round: position the connector with the keyway (hollow in the centre) towards the solder side (Figures 5 & 6), i.e. with the flat side towards the component side.

All that now remains is to program the PIC with its USB firmware (file: nunchuk.hex). There are two options for doing this: buy the PIC pre-programmed from Elektor [1], or else use a universal programmer if you have a blank microcontroller.

The nunchuk.hex file contains the bootloader plus the example program `hid_clavier_trame.fcf` (see program examples below).

There's no driver to install as the PC sees the interface as a keyboard belonging to the class of HID peripherals: these drivers are included

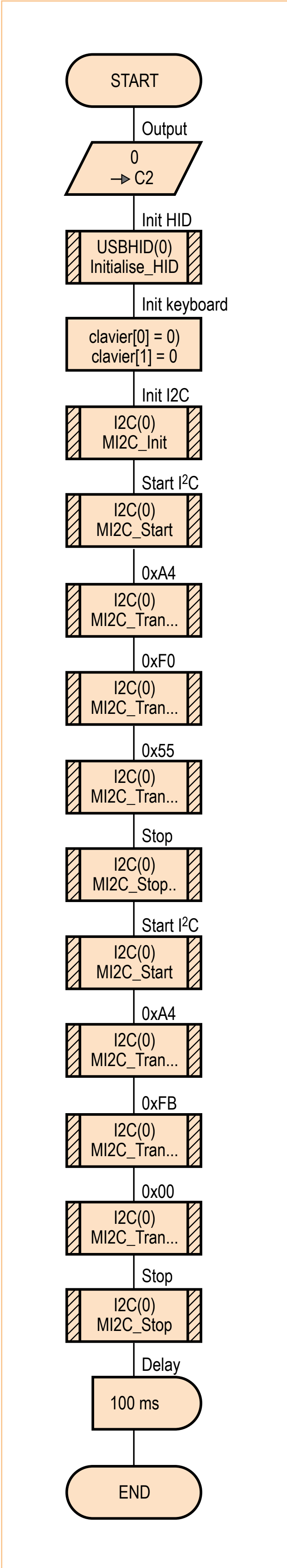


Figure 3. Initializing the Nunchuk controller.

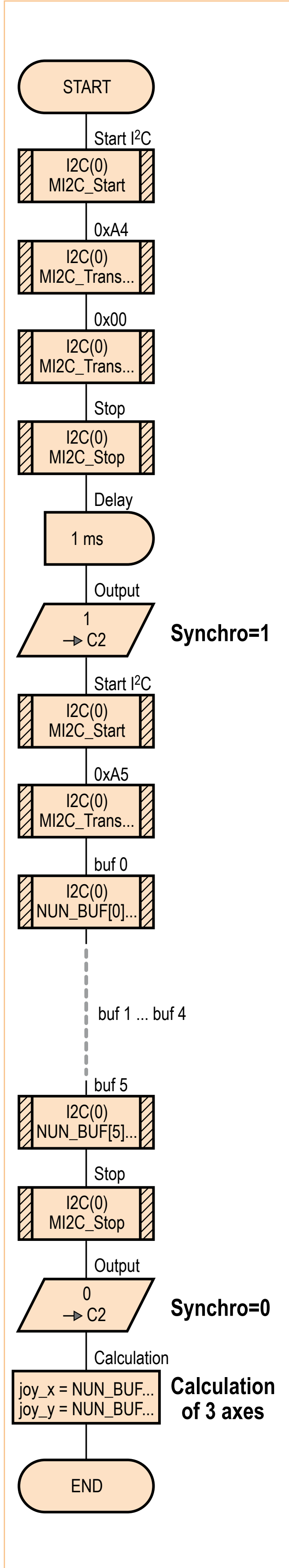


Figure 4. Reading the data from the Nunchuk controller.

in the operating systems, so the interface is compatible with Windows, Linux, MacOS, etc. with no installation needed.

Software

All the programs provided have been produced using Flowcode v4. This software is ideal for programming the 18F2550, as even a beginner in computing will be able to handle the HID (human interface device) drivers for the USB link and for the I²C bus.
Those used to using C will have more difficulty porting the programs, but will have to remember to tell the compiler to start the program from the address $800 in order not to delete the bootloader if they want to go on using the USB program (**Table 7**).
Table 7

Using the bootloader

At power-up...
To program a Flowcode hex file, all you have to do is hold the 'boot' button pressed as you apply power to the board. The LED D1 flashes

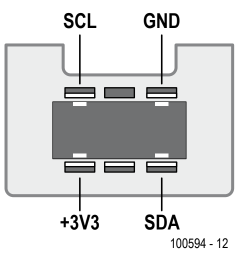


Figure 5. The Nunchuk connector, showing the keyway (hollow), which must be oriented towards the solder side of the board.

Figure 6. The two slots made with a min disc-cutter are vital so as to be able to plug the controller cable in.

Table 7.

Address	
	Flash memory
	user program
	Flowcode
$0800	
$07FF	bootloader
$0000	(do not delete)

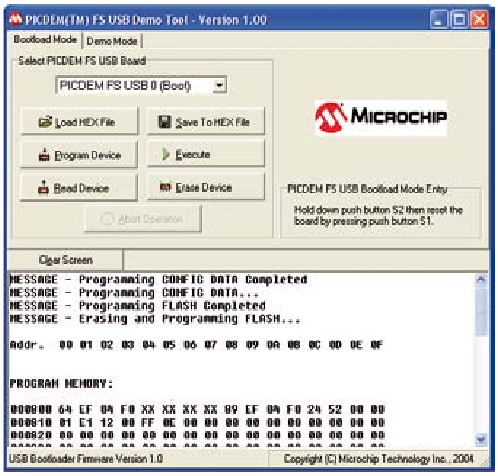


Figure 7. Running the PDFSUSB.exe programming software that lets you access the contents of the 18F2550's Flash ROM.

to indicate that the drivers must be installed.
The various files are contained in the 'USB Framework' pack from Microchip [3]. We strongly recommend you opt to install it in the default directory. You'll find the drivers in the `\Microchip Solutions v2010-10-19\USB Tools\MCHPUSB Custom Driver\MCHPUSB Driver\Release` directory. They can be downloaded from the Elektor website [1].

After installing the driver, run the programming application PDFSUSB.exe which is located in the `\Microchip Solutions v2010-10-19\USB Tools\Pdfsusb` directory.
Select the PICDEM FSUSB 0 (Boot) driver.
Load a hex file, click on 'program device', then 'run' (**Figure 7**).

Example programs

I'm providing four examples of applications [1] which will be easy to modify so you can use them elsewhere or in a different way.
hid_clavier_trame.fcf (basic program included in the firmware)
This program lets you test that everything is working properly.

When you press button Z on the Nunchuk, the digital values for all three axes are sent over the USB link. They will be displayed in Notepad, then the interface board emulates a PC keyboard. When button S1 is held down, only the digital value for the X axis is sent to the PC.
You can then analyse these data in a spreadsheet, as shown in **Figure 8**.

hid_joytick.fcf
The Nunchuk turns into a joystick on the X and Y axes. The joystick calibration can be accessed in the Windows 'Settings' ('Games controllers'). Warning: if you don't calibrate it, the joystick operation will not be optimum.

hid_souris_accel.fcf
('souris' = mouse) Turns the Nunchuk controller into a computer mouse: X axis = left/right movement and Y axis = up/down movement. Z & C buttons: left/right click You're likely to be a bit disconcerted at first by the sensitivity of the controller.

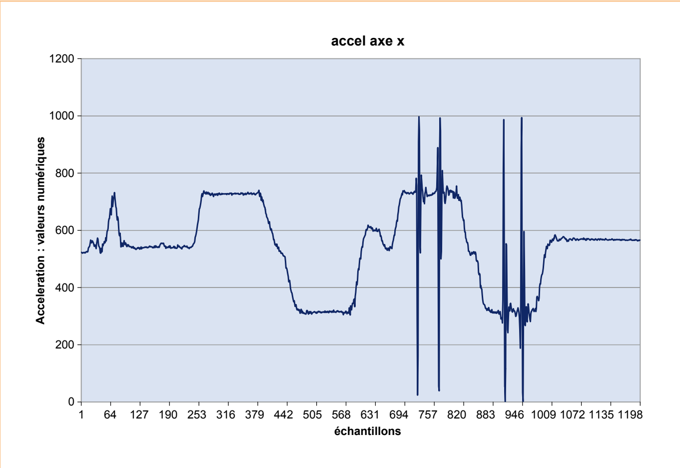


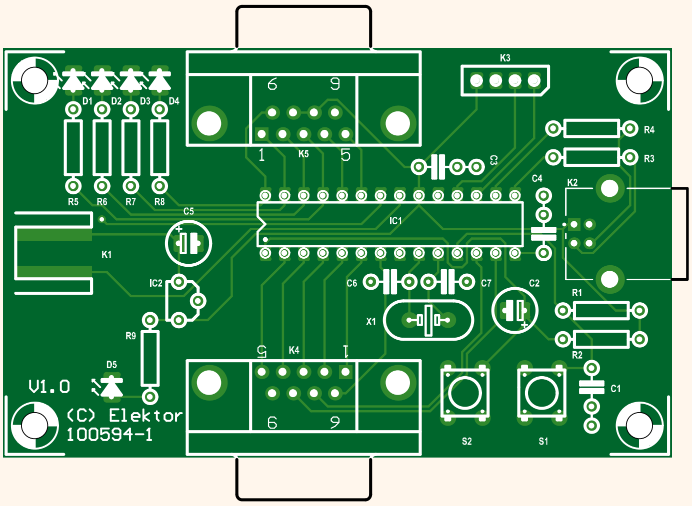
Figure 8. The 'boot' button also lets you send the accelerometer X-axis values, which you can then analyse in a spreadsheet.

COMPONENT LIST

Resistors
R1,R2 = 10kΩ
R3,R4 = 22Ω
R5–R9 = 1kΩ

Capacitors
C1 = 470nF
C2 = 10µF 16V radial
C3 = 100nF
C4 = 220nF
C5 = 2.2µF 16V radial
C6,C7 = 22pF

Semiconductors
D1–D5 = LED, 3mm, low current
IC1 = PIC18F2550-I/P, DIP, 28-pin, programmed, Elektor # 100594-41
IC2 = LP2950ACZ-3.3



Miscellaneous
S1, S2 = pushbutton
K2 = USB-B connector, PCB mount
K3 = 4-pin pinheader
K4, K5 = 9-way sub-D socket, right-angled pins, PCB mount
X1 = 20MHz quartz crystal
PCB # 100594-1

nunchuck_dmx.fcf
The Nunchuk turns into a ‘moving head’ stage spot controller using the DMX protocol on the serial connector K3 (RC6: TXD). Warning! You will need to modify the program depending on the spot being used, and add an RS485 bus IC: 75176.

(100594)

Internet Links

[1] www.elektor.com/100594

[2] http://wiibrew.org/wiki/Wiimote/Extension_Controllers#Nunchuk

[3] Microchip USB FrameWork: http://ww1.microchip.com/downloads/en/DeviceDoc/MCHP_App_Lib_v2010_10_19_Installer.zip
Update list:
www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2896

[4] www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010280

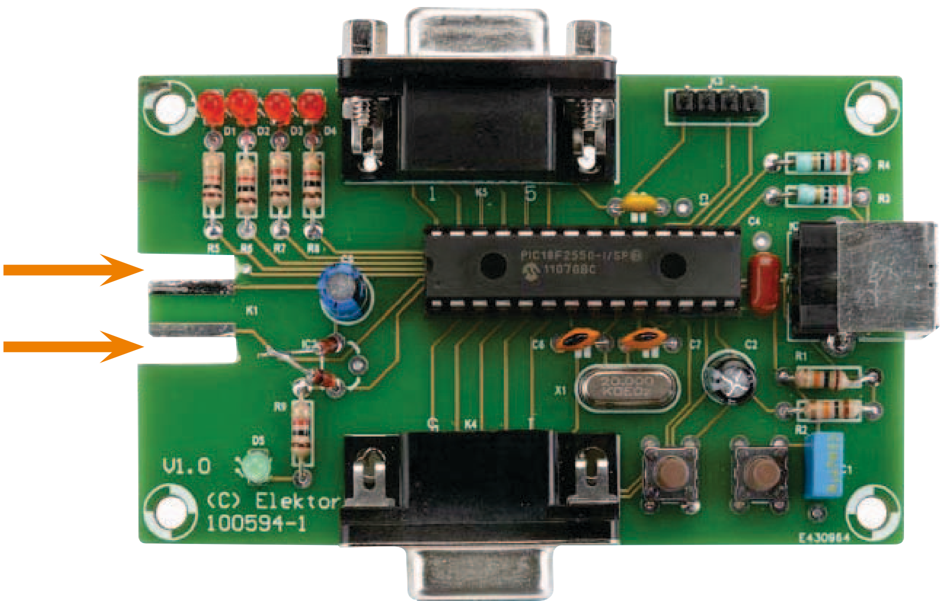
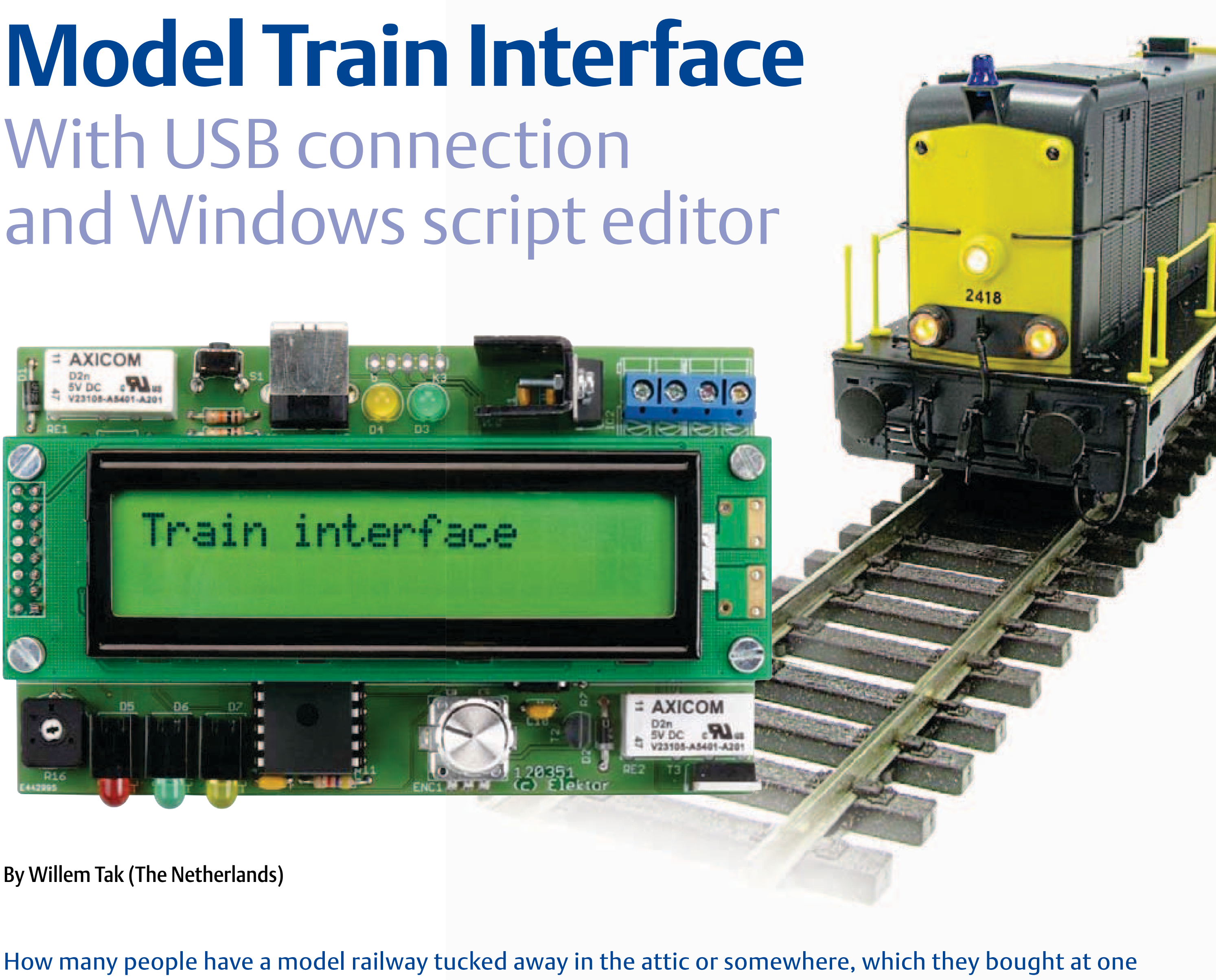


Figure 9. Initially, there are no slots in the PCB either side of K1 for the Nunchuk plug. The serial interface (K3) is used only by the nunchuck_dmx.fcf program and any personal applications you may think of. At the time of building the prototype, the 3.3 V regulator drawer was empty, which explains why there are two diodes in place of IC2.

Model Train Interface

With USB connection and Windows script editor



By Willem Tak (The Netherlands)

How many people have a model railway tucked away in the attic or somewhere, which they bought at one time for themselves or the children? After going around and around for a while the fun usually starts to wear off and there are very few who will continue to expand and automate their railway. With this small circuit your model railway will have a few additional features and more intelligence, without the need to buy intelligent trains and other expensive model railway equipment.

Every year, during the month of December, the lady next door builds a beautiful Christmas landscape. To add some life to the scenery a train runs through it. However, this train going around forever and ever in circles gets to be a little boring after a while, so something had to be done about this. Could this little train not change direction or speed every once in a while? And is it possible to stop at the station sometimes?

There are obviously countless methods to realise this, such as the excellent EEDTS (EDiTS) from Elektor, but this simple application begs for an equally simple solution. The idea is to write a script which contains a sequence of instructions (drive forwards or backwards at a particular speed, stop for a number of seconds, drive to the station, etc.) and to have this script executed by a circuit specifically designed for this purpose.

Hardware

This circuit has been designed for use with model trains that operate on DC. It has been tested with a Fleischmann 9336.

At the heart of the circuit is a Microchip PIC 18F4550 (see **Figure 1**). This micro was selected because of its built-in USB interface, which we can use nicely to communicate with the script editor running on the PC. This is very easy: you write a script using Windows and then send it to the interface. Because of the USB functionality, using a 24-MHz crystal is absolutely necessary.

To operate the circuit, a two-line LCD was chosen, together with a rotary encoder with pushbutton. The LCD is controlled directly by the PIC. Data line RD7 is fitted with a pull-

down resistor (R14) because the software makes use of the Busy flag.

The power supply design with relay Re1 appears a little strange at first, but this has to do with the circuit really having two functions: it is a USB interface **and** a train controller. When the power supply relay is in the non-energised state, the 5-V power supply voltage of the USB interface is used to power the circuit. This then functions as a HID-USB device and in this state scripts can be read and loaded. When an external power supply of 15 V is switched on, the relay is energised and the controller gets powered via a 7805 regulator (IC2).

The pulses that drive the train are available at output RB4 of the controller. This is a pulse-width modulated signal with a frequency of 100 Hz. The train goes slow with narrow pulses and with wide pulses it ‘flies’ along the tracks. The 5-V pulses from the controller are ‘enlarged’ by T1 to 15 V, after which Schmitt-trigger gates IC3.A to IC3.C are used to enhance the edges. The pulses are subsequently sent to an N-channel MOSFET type FQP12N60. This FET can cope with quite a lot, but the author is no model railway expert. I assume however that there are model trains requiring much more current than the one with which the circuit was tested. The type selected here can carry quite a bit of current (about 10 A), but usually no more than about 1 A is required. Forwards and backwards driving directions are accomplished with the driving relay (Re2). This simply reverses the polarity of the voltage on the tracks. You will have to experiment which way around the wires have to be connected to the tracks. If the program indicates that the train should travel forwards but it actually goes in the opposite direction then you have to swap the wires.

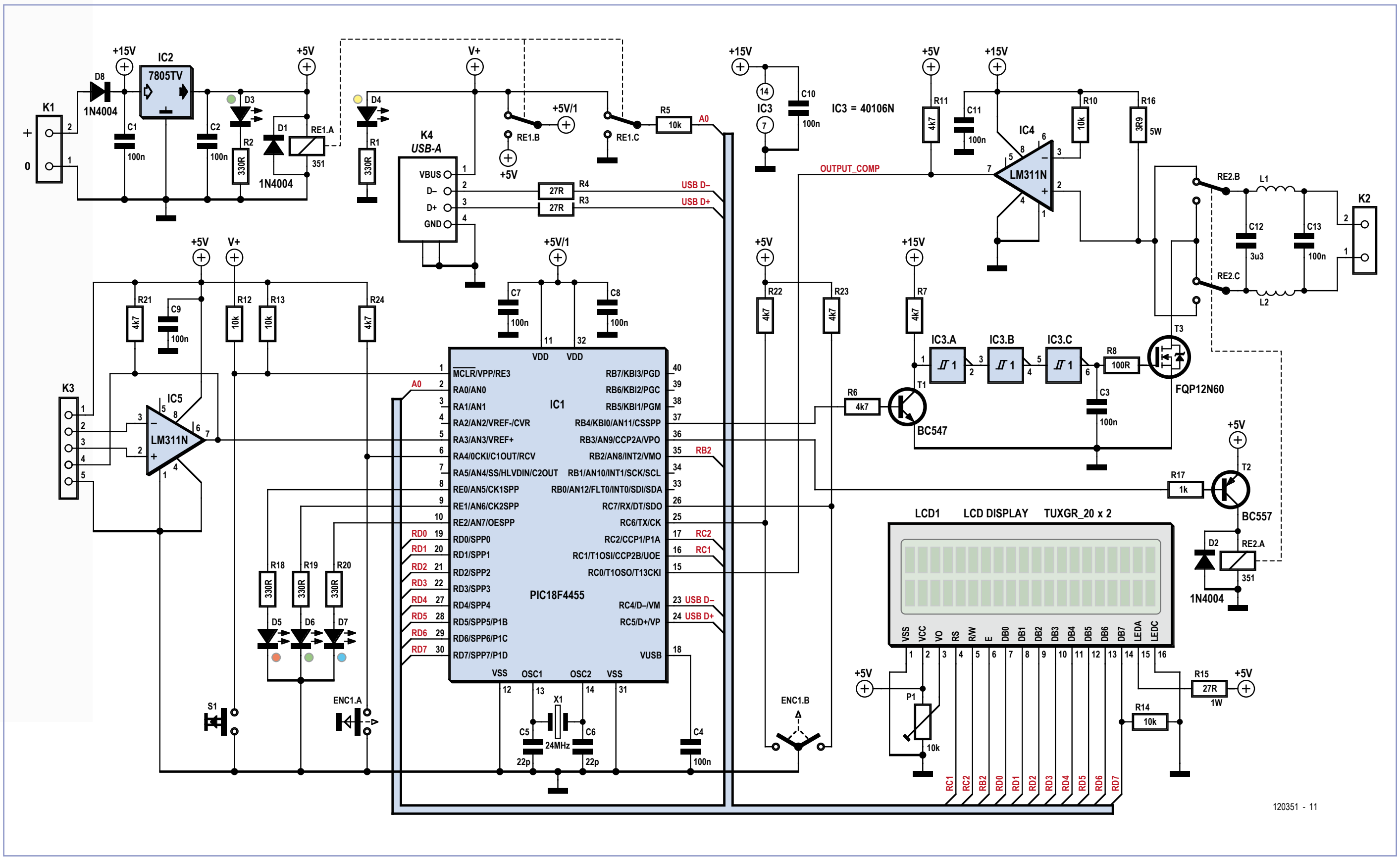


Figure 1. The controller generates a pulsewidth modulated signal, which drives the train via MOSFET T3.

Since the little motor in the train (a substantial inductance!) can generate sizeable noise spikes, there is a suppression network (C12/C13/L1/L2) between the relay contacts and connector K2. Its purpose is to prevent these spikes from upsetting the control electronics. Capacitor C3 between the CD40106 and the MOSFET contributes to this as well; the edges of the control signal are a little less steep as a result.

Via opamp IC4 there is a feedback signal for the controller (pin 15), so that the control-

LED indications

Internal LEDs (D3/D4):

• Yellow (D4)	on when USB is in use
• Green (D3)	on in Train mode with external power supply

Front-panel LEDs (D5/D6/D7), USB Mode:

• Blue (D7)	flashes during initialisation of the USB Mode
• Blue (D7)	on when the program trein.exe is running
• Blue (D7)	can be controlled by the program trein.exe

Front-panel LEDs (D5/D6/D7), Train mode:

• Red (D5)	on in Train mode when the train is moving
• Red (D5) + Green (D6)	on when the train is stationary during the command 'wait'
• Green (D6)	on when the train is stationary at the station
• Blue (D7)	on when the train is stationary when the station has not been found
• Red + Green + Blue	on when there is an LCD error
• Red + Blue	on when there is an EEPROM error
• Running lights	train is not moving according to the script

ler can check that the instructions from the script are actually being carried out; when the train is moving, IC4 receives pulses at its + input.
Station detection is achieved with the aid of a light barrier (separate schematic in **Figure 2)**. This is a simple U-shaped construction consisting of two upright wooden posts, one on each side of the rails. An LED is mounted in one post and an LDR on the other. When the train passes through the barrier it interrupts the light from the LED that shines on the LDR. The trigger level can be adjusted with a 10-kΩ potentiometer, which is connected to K3.

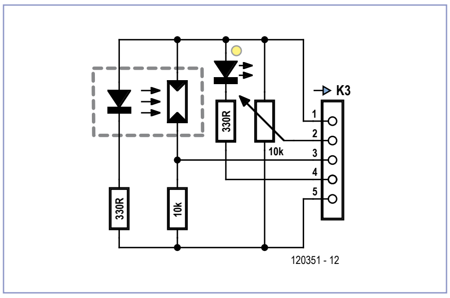


Figure 2. These parts are mounted near the light trap and are connected to K3 on the circuit board.

The various LEDs on the board indicate the different power supply and communications states. When the circuit is powered via the USB cable the yellow LED D4 is on. When the USB connection has been established the blue front panel LED D7 will flash. Once the program Trein.exe is running on the PC, the blue front panel LED is on continuously. When an external 15-V power supply voltage is connected the green LED D3 is on and the red front-panel LED D5 flashes briefly to indicate initialisation. For a complete overview see the sidebar 'LED indications'.

Software

The software for the 18F4550 was written entirely in assembler, the PC program to make the scripts is written in Visual Basic. The assembler code consists of two parts: the USB-HID interface and the control of the train.

USB code

The USB code is a standard HID interface for the PIC18F4550. It has a VID/PID of 0D59/5275 built in. After a successful enumeration, the standard UDB loop is started, set to 6 ms. The USB interface communicates with the PC via a buffer that is 64 bytes in size. Via this buffer the scripts from the VB environment are loaded into the 4550. The task of the USB program is to store and

Byte composition

The script supplies two bytes per step. They are coded as follows:

• High byte bit 7:	1 when driving forwards
• High byte bit 6:	1 when driving backwards
• High byte bit 5:	1 when searching for the station
• High byte bit 4:	1 during a wait action
• High byte bit 3:	service bit
• High byte bit 2–0:	train speed (1–7)
• Low byte:	time (1–255 s)

The service bit (bit 3) has multiple functions, depending on the byte in which it is used.

• In byte 0:	Toggle bit for EEPROM Write action
• In byte 2:	A '1' indicates script1
• In byte 4:	A '1' indicates script2
• In byte 6:	A '1' indicates script3
• In byte 8:	Status of blue USB front panel LED

read the scripts from the EEPROM. Three scripts can be used.

The instructions as well as the data come from the VB program. A format was selected comprising of 32 instructions per script (which turns out to be sufficient in practice), because this fits nicely in the 64-byte buffer of the HID. There is also still some space for a few control bits, because the instruction byte for a command is only seven bits in length. The sidebar 'Byte composition' details the layout of the bytes.

As usual the USB program takes care of the timing with the USB connection to the PC. To prevent the program from getting into trouble with the slow write times of the EEPROM (4 ms according to Microchip), only one byte is written to the EEPROM for each USB loop.

The entire write operation is initiated by a Toggle byte in the VB environment and a few flags in the USB code. One byte is then written for each of the next 64 consecutive cycles. During this period the PC indicates it's waiting and no data can be entered.
The USB program never reads directly from the EEPROM (except when starting up), but ensures that the USB-in-buffer is always provided with the latest state of the actual script via separate buffers in the PIC.

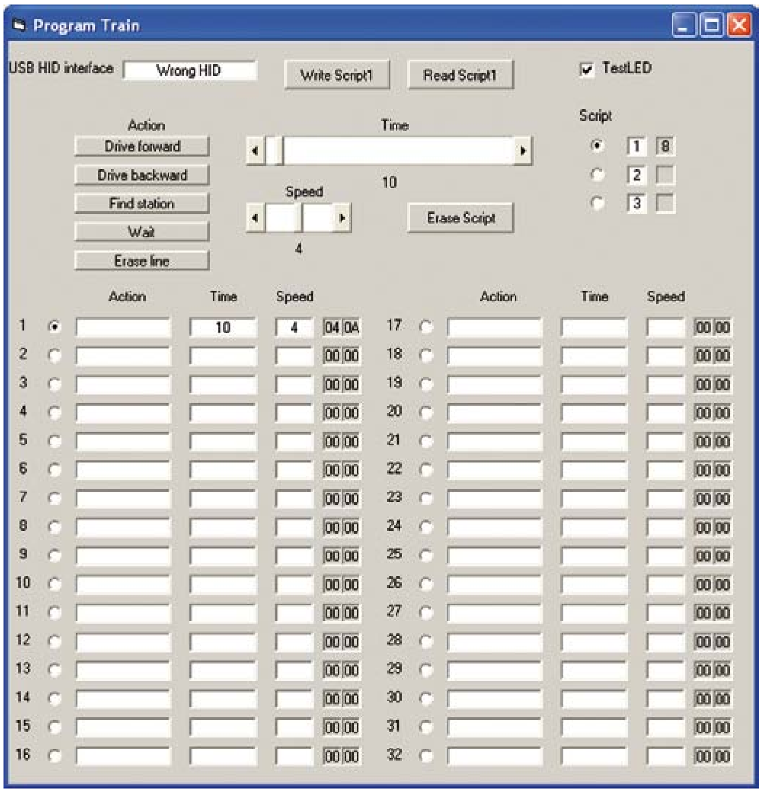


Figure 3. With the aid of the corresponding Windows program it is easy to enter the movement pattern of the model train.

Train code
The second initial operating mode for the PIC is the Train mode; this selection is made via an input (A0) on the PIC.

The train interface communicates with the user via a two-line LCD (contrast adjustment using P1). Using the rotary encoder it is possible to scroll through the various instructions. The pushbutton on the encoder is used to confirm a command.
The assembly code for the train controller is quite simple. A line from the script needs to be retrieved and then needs to be carried out. In most cases the routine that supplies the pulsewidth modulated signals is called. This is controlled via a timer which determines how long the action should last. The progress of these steps is always visible on the LC display.

In Train mode the menu program is started first. This can start up one of the three scripts or allow the train to be controlled manually. The settings can be reached using the reset button.

The heart of the program is the DRIVE routine. This runs at 100 Hz via normal timing loops (i.e. not using an interrupt) and controls the train with a variable pulsewidth with a frequency of 100 Hz. The on- and off-times of the pulsewidth timing come from tables and depend on the minimum pulsewidth that has been set. The routine also supplies a seconds pulse for the timing of various actions.
The rotary encoder can be read by two different routines with different sensitivities (fast and secure). The pushbutton for starting/interrupting is part of the encoder.

Putting scripts together

The purpose of the Windows program Trein.exe (**Figure 3**) is to compose the scripts which are then carried out by the train interface. The order of connecting things is important when working with this program. First the USB cable (of a running PC) needs to be connected. (The very first time that this is done it is possible that Windows generates an error message, but usually the device is nevertheless correctly installed. Each subsequent time will then be without problems.) Subsequently the program Trein.exe is started. The blue front-panel LED D7 will light up and the program shows in the box 'USB HID interface' the message 'Treininterface'. The correct operation of the USB-interface can be tested with the button 'TestLED'. Ticking and un-ticking of this checkbox activates or extinguishes

the blue front-panel LED. The program can be used to compose or modify scripts, read from the memory in the train interface and put them back into the memory.
There are three scripts, each of which can be up to 32 lines long. Each line can be used to program one of the following actions: drive forward, drive backward, find station and wait.
The program is operated entirely using the mouse (and therefore not always entirely logical). Each of the four actions mentioned has a corresponding time variable which has to be set with a slider between 1 and 255 seconds. This time determines the duration of the action. With the exception of 'waiting' the actions also have a speed setting. This can be set with a second slider between 1 and 7. This number indicates the speed of the train during that action, where 1 is the slowest and 7 is the fastest.
The operation is simple: choose a script, select a line, select an action or time which will then appear on that line.
The end of the script is indicated with a blank line. If all 32 lines have a function defined then the program will continue on line 1 after completing line 32.
The exception to this is script 1. If all 32 lines are used then the program will continue with the first line of script 2. Should script 2 also contain 32 lines then execution continues with line 1 of script 3. It is therefore possible to make a script with a maximum of 96 steps.
The program has buttons to retrieve the scripts from the memory (Read Script) and to write the current script to the memory of the controller (Write Script). Note that these write operations do not require any further confirmation but are carried out immediately. During the Read- and Write operations the program cannot be used for a short time; this is indicated by a 'wait' window.
The program also shows a few windows that are greyed out. These contain the values which will be written into memory and are only of interest to the programmer.

Construction

The Elektor lab designed a printed circuit board for the train interface, containing the entire control including the power stage (**Figure 4**). The construction is not difficult, only leaded components have been used. Voltage regulator IC2 is fitted with a small heatsink. The MOSFET is fitted near the edge of the board so, if necessary, it can also be bolted to a (small) heatsink.
Once the board has been fully assembled and has once more been visually checked, you can fit the programmed microcontroller (available from Elektor, if you are unable to program it yourself) in its socket and then the display can be mounted on the top of the board using four 20-mm long stand-offs. You can make the connections between the display and the board using short pieces of wire or a 2x8-pin PCB connector and corresponding header with extra-long pins.
The board can now be connected to the PC with a USB-cable and you can start to write a script. You can then load it into the EEPROM of the controller.

Usage

Connecting the train interface to the model railway is simple. Connect to K1 an AC power supply with an output voltage of 15 V and capable of supplying a current of one or more amps (depending on the train used) and connect K2 with the tracks of the model railway. Using the coils listed for L1 and L2 in the parts list, the circuit can supply a current of up to 1 A. This is more than enough for the average model train.
The parts shown in **Figure 2** you will need to mount yourself to make the light barrier. After switching on the 15-V power supply the circuit starts in Train mode. The red front-panel LED will turn on and the operation is now via the rotary/pushbutton below the LCD.
After the welcome message, the rotary switch can be used to select one of five options. These are script1, script2, script3, manual control and test scripts. When the pushbutton is pushed the selected option is activated.
The three script options start the corresponding script.
The LCD shows on the top line the current instruction from the script and at the same time the sequence number of this instruction. The bottom line indicates the actual action (including direction of travel) and the time (in seconds) remaining for this action. Driving forwards, backwards as well as waiting are unambiguous; the station search action requires a little explanation (for more information: see the additional description on the Elektor website). This really consists

COMPONENT LIST

Resistors

R1,R2,R18,R19,R20 = 330Ω
R3,R4 = 27Ω
R5,R10,R12,R13,R14 = 10kΩ
R6,R7,R11,R21–R24 = 4.7kΩ
R8 = 100Ω
R15 = 27Ω 1W
R16 = 3.9Ω 5W
R17 = 1kΩ
P1 = 10kΩ preset

Capacitors

C1–C4,C7–C11,C13 = 100nF
C5,C6 = 22pF
C12 = 3.3µF MKT, lead pitch 15mm

Inductors

L1,L2 = choke, 330µH, 0.9A, 0.32Ω (e.g. Panasonic ELC10D331E, Farnell # 1749073)

Semiconductors

D1,D2,D8 = 1N4004
D3,D6 = LED, green, 5 mm
D4 = LED, yellow, 5 mm
D5 = LED, red, 5 mm
D7 = LED, blue, 5 mm
T1 = BC547
T2 = BC557
T3 = FQP12N60
IC1 = PIC18F4455-I/P (programmed, Elektor # 120351-41)
IC2 = 7805
IC3 = 40106
IC4,IC5 = LM311

Miscellaneous

RE1,RE2 = PCB relay, 5V (e.g. V23106-A5401-A201)
LCD1 = LCD 2x20 characters with backlight (e.g. Midas MC22005A6W-SPTLY)
ENC1 = rotary encoder w. pushbutton (e.g. Bourns PEC11-4230F-S0024)
K1,K2 = 2-way PCB screw terminal block, pitch 5mm
K3 = 5-pin SIL pinheader
K4 = USB-B connector, right angled, PCB mount
S1 = pushbutton (e.g. Omron B3F-3100)
X1 = 24MHz quartz crystal
PCB # 120351-1 (see www.elektor.com/120351)

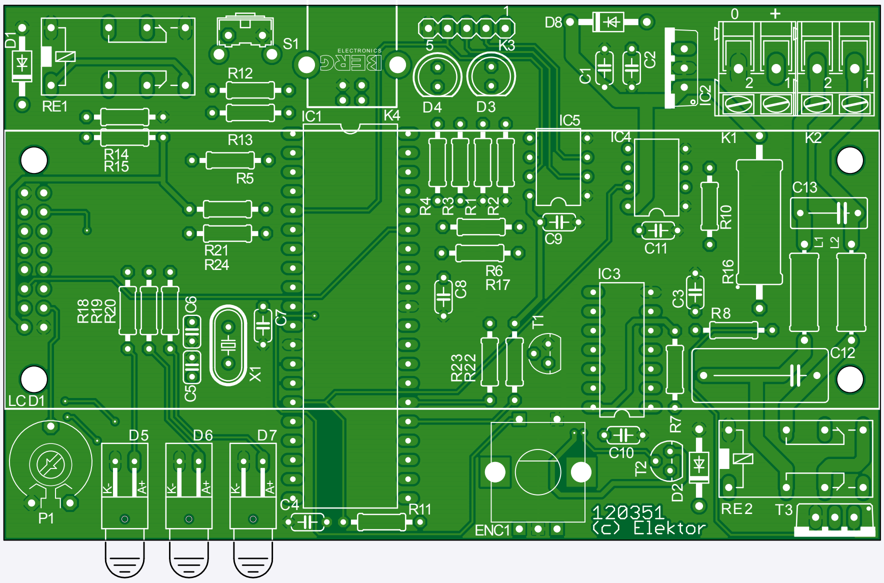


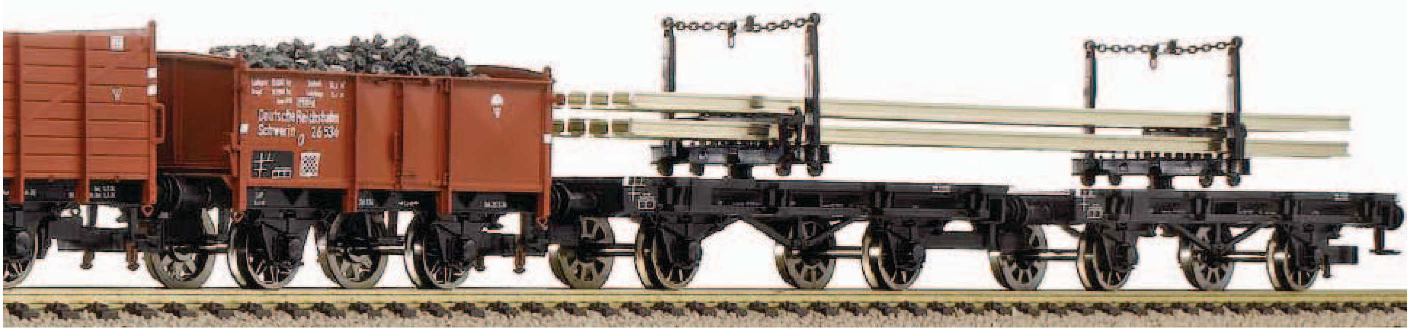
Figure 4. The dimensions of the PCB are such that the display can be mounted above it using four stand-offs.

of two instructions: firstly the train has to pass through the light barrier that marks the station. During this action the maximum time is indicated. That is because the actual travel time is not known, since it depends on the location of the train and its speed. Once the station has been found the waiting time assigned to the station line commences.

A script can be interrupted at any time using the pushbutton.

In addition to the PC-software and the firmware for the microcontroller the Elektor website at www.elektor.com/120351 also supplies supplementary information about the train interface.

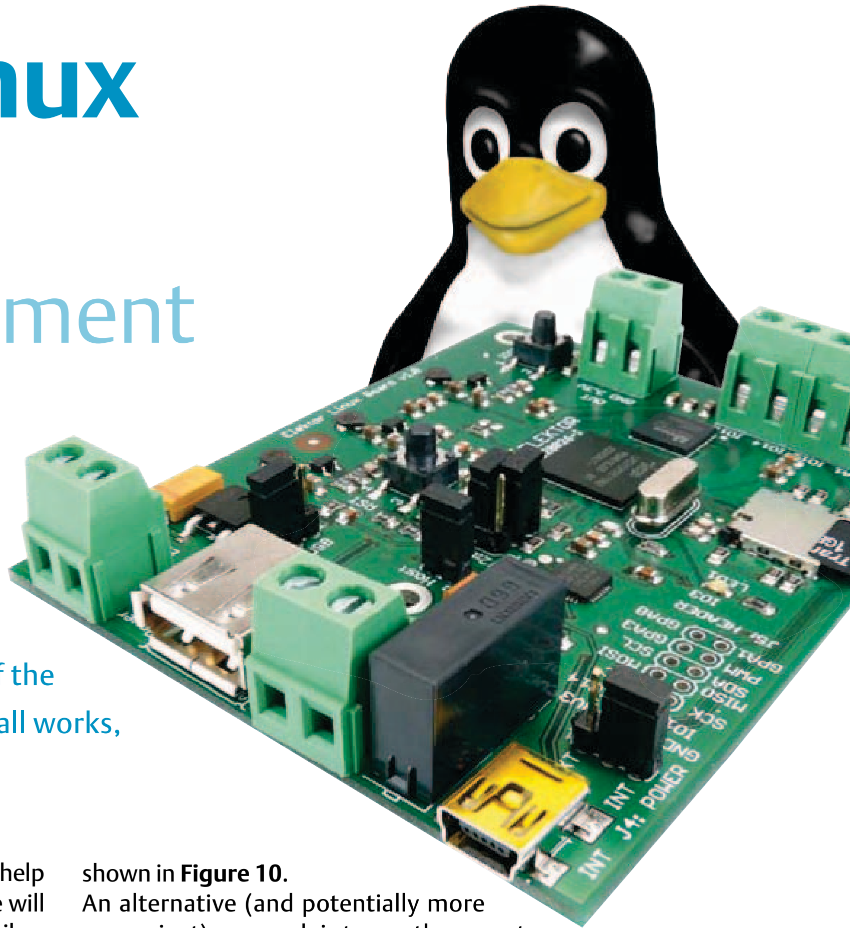
(120351-I)



Embedded Linux Made Easy (3)

Software development

By Benedikt Sauter [1]



It takes the right software to bring a microcontroller to life. Beyond the usual firmware, in an embedded GNU/Linux system we have to deal with building the components of the operating system. In this article we show how it all works, and even write our first program in C!

It is easiest to develop for an embedded Linux system with the help of a conventional Linux system, normally running on a PC. We will base our experiments on version 12.04 of the 'Ubuntu' [2] distribution. What do we need to install such a system? Not a lot: a little free space on the hard disk and, ideally, a network connection.
The first thing to do is download the image of the installation CD from the internet [2]. We can use either the 32-bit or the 64-bit desktop variants: if in doubt, select the 32-bit variant.
Once the CD image is downloaded it has to be burned onto a CD using a suitable program. It is important to burn the file to the disk as an 'image' rather than copying it as a simple file.
Now insert your newly-burned CD into the PC and boot from it (which may require some adjustments to your BIOS settings). Ubuntu starts up with the splash screen shown in **Figure 1**. To make the system more convenient to use will we install it to the hard disk, by selecting the second menu option (see **Figure 2**). You will now be guided step-by-step through the installation process. First choose your language (**Figure 3**). We do not need to install any third-party software (**Figure 4**). The next window, shown in **Figure 5**, lets you choose to let Linux occupy the whole hard disk (which would be suitable for a machine that does not already have another operating system installed); alternatively, you can install Linux to a second hard disk, or partition the main hard disk into two areas, one area retaining the already-installed operating system and the other dedicated to the new installation. In **Figure 6** you select the drive or partition that will be used, and then proceed as shown in **Figure 7** and **Figure 8**. We will explain more about the password that you are asked to set (**Figure 9**) later. If all goes well, you should reach the point shown in **Figure 10**.
An alternative (and potentially more convenient) approach is to run the operating system in a virtual machine. The author has prepared an image of a Linux computer set up for development especially for *Elektor* readers, and it can be downloaded from the *Elektor* website [3]. The virtualisation program 'VirtualBox' is needed to run the image: it can be downloaded free of charge at [4]. When VirtualBox has been installed, the image is loaded by simply selecting 'File > Import Appliance' from the main menu; it can then be run immediately.

Toolchain on CD

With the new operating system running the next step is to install the toolchain. If you are using the VirtualBox image this has all already been done for you, and you can skip the next two sections.

The quickest way to do things in Linux is usually to use the console. Simply open up a new terminal window on the PC by pressing Con-

Figure 1. Use down-arrow to select the second menu item.

Figure 2. Select 'Install Ubuntu' and press Enter.

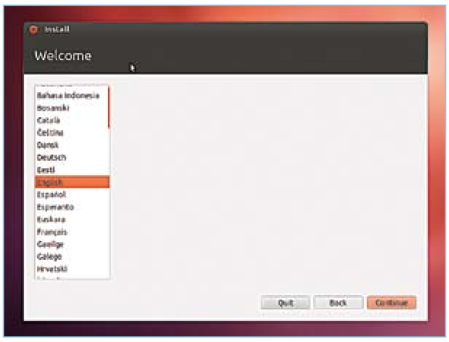


Figure 3. Choosing your language.

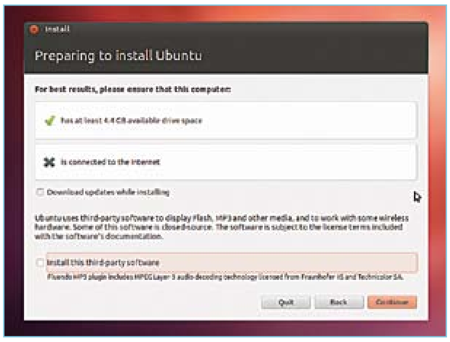


Figure 4. Preparing for installation.

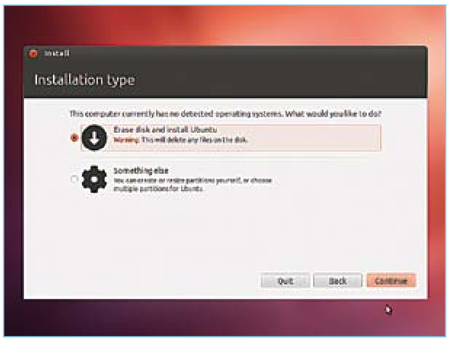


Figure 5. Choosing where the operating system will be installed.

trol-Alt-T. In the terminal, switch to the directory '/tmp'

```
cd /tmp
```

and then download the ARM toolchain CD directly using the 'wget' command thus:

```
wget ftp://ftp.denx.de/pub/eldk/5.0/iso/armv5te-qte-5.0.iso
```

This will download a CD image. The image can be opened directly using Linux: there is no need actually to burn the file to a CD. However, the file does need to be 'mounted', which makes it visible as a set of files to the operating system. First switch the the directory '/media':

```
cd /media
```

Figure 6. Selecting the hard disk.

In theory you can mount the contents of the CD image wherever you like in the file system, but there are certain conventions in the Linux world that make it easier for people to find their way around a new system. We will look in more detail later at the standard arrangement of the file system.

We now want to create a new empty directory called 'eldk-iso', where we will subsequently mount the CD image. Although this might seem odd to someone familiar with Windows, we are not going to copy the files from the CD image to the new directory: instead, the new directory just marks the place in the file system from which the contents of the CD are made accessible. In Linux, everything is handled through files and directories.

Here is the command to create the new directory:

```
sudo mkdir eldk-iso
```

Figure 7. Selecting the time zone.

The machine will prompt you for a password, as creating the directory requires you

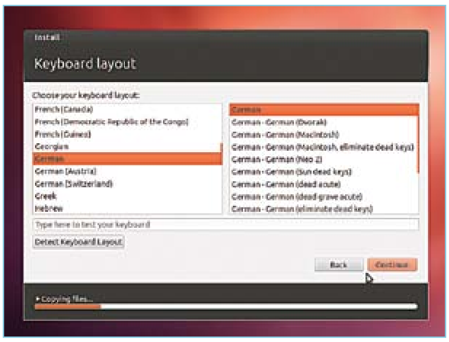


Figure 8. Selecting the keyboard layout.

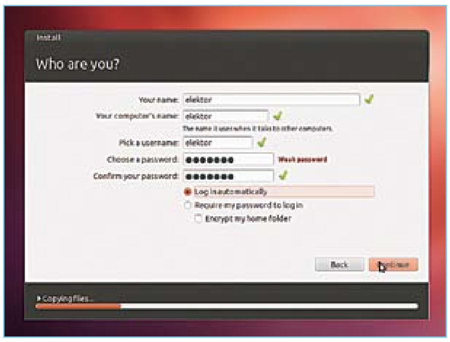


Figure 9. Entering a user name and password.

Figure 10. The installation process begins.

Permissions and privileges

A Linux system always has a user called 'root'. This user has the highest level of privileges on the system: other users typically only have ordinary privilege levels. Full access to system files, devices and so on requires 'root privileges'. One approach is simply to execute all commands as the user 'root' but this is not advisable. One of the reasons Linux is so secure is that the restrictions on what ordinary users can do prevent a lot of potential damage. Users are normally only given permission to run the programs they need, not to access system files or other important information.

The Linux programmer can, however, grant any application a higher level of privileges as needed, and so there is no need to execute all commands as the user 'root'. We will also create a new user account on our Linux board for carrying out ordinary tasks and running the programs we write.

If a user briefly requires root privileges, for example to create a directory within a system directory, then we can use the command sudo provided in most modern Linux distributions (including the version of Ubuntu we are using) as a prefix to the command proper. This indicates that the command is to be run as if by the root user.

You will be prompted for a password. If you have installed the Linux system on the PC yourself, you will have set up this password as part of the installation process; if you are using the VirtualBox image, the password is 'elektor', entirely in lower-case.

(briefly) to have 'root privileges': see the text box 'Permissions and privileges'. To make this happen, we have prefixed the normal command with 'sudo', which asks the user for the password that was configured when the system was installed. If you are using the VirtualBox image, the password is 'elektor', entirely in lower-case.

The CD image can now be mounted in the file system to allow us to access its contents:

```
    sudo mount -o loop /tmp/armv5te-qte-5.0.iso /media/
eldk-iso
```

From our current position in the file system we can change directory into the CD image by typing:

```
    cd eldk-iso
```

Installing the toolchain

In the directory you will find a small script that you can use to install the toolchain. Again, you need root privileges to install new programs:

```
    sudo ./install.sh -s -i qte armv5te
```

The following message should appear:

```
    *** Installing ./targets/armv5te/eldk-eglibc-i686-
arm-toolchain-qte-5.0.tar.bz2
```

When installation is complete you can leave the CD directory:

```
    cd ..
```

This step is important, as the next thing we will do is unmount the CD image. This will not work if you are currently in a directory within the image: the operating system will refuse to execute the command and an error message will be printed.

```
    sudo umount /media/eldk-iso
```

The directory you created for the toolchain CD can now be deleted.

```
    sudo rmdir eldk-iso/
```

When installation is complete the toolchain will be located in the directory '/opt/eldk-5.0/'. So that we can access the programs in the toolchain using the command line from whichever directory you happen to be in, we have to add this directory to the PATH variable. This is a Linux 'environment variable' which contains a list of directories in which the system will automatically look for programs to execute.

The best approach is to write a small script file (call it 'set.ch') which you can run from the console before using the toolchain. Create a new file using the editor:

```
    gedit set.sh
```

and add the following lines to it:

```
#!/bin/bash

    P1=/opt/eldk-5.0/armv5te/sysroots/i686-oesdk-linux/
usr/bin/armv5te-linux-gnueabi/
    P2=/opt/eldk-5.0/armv5te/sysroots/i686-oesdk-linux/
bin/armv5te-linux-gnueabi/
```

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
export PATH=$P1:$P2:$PATH
```

The last command here adds the path mentioned above to the PATH variable.
Then write the new file to your start-up directory (called the 'home directory'). For the commands in the file to take effect, you have to cause the shell (the Linux command line interpreter) to read them in (or 'source' them). One way to do this is with the following command:

```
. ./set.sh
```

Type this carefully: the line starts full stop, space, full stop!

If you would prefer not to have to type this command every time you bring up a new console you can include it in the file '.bashrc', which is automatically executed whenever the shell starts up. The file is located in your home directory, which you can switch to using the command `cd` without any arguments. You can edit the file using the command:

```
gedit .bashrc
```

The compiler in action

The toolchain programs that we will use to build the Linux kernel and the bootloader all have names that begin with 'armv5te-'. For example, the GCC compiler is called 'armv5te-gcc'. Typing the command

```
armv5te-gcc --version
```

will give the version number of the compiler (note that there are two dashes before 'version'). If this command works, it means that you have successfully set up the path to the toolchain programs.
To compile application programs for Linux we need to use the toolchain commands that start 'arm-linux-gnueabi-' rather than 'armv5te-'. So how do we compile a simple 'hello world' program?

First create a source file

```
gedit hello.c
```

with contents as follows:

```
#include <stdio.h>

int main(void)
{
  printf("Hello world!\r\n");
  return 0;
}
```


Save it and leave the editor. Now we are back in the console. To compile the program, type:

```
arm-linux-gnueabi-gcc -o hello hello.c
```

To test whether the above process has been successful, we can copy the file 'hello' that the compiler has created to the *Elektor* Linux board's SD card. Make sure the board is off and remove the card. Insert it into the PC's card reader, and plug the reader into the PC. Wait a few seconds for the machine to detect the card. Normally Ubuntu will automatically pop up a window when this happens: since we will be copying to the card using the console, we can close this window.

Full-scale operating systems such as Ubuntu automatically mount external storage devices when they are plugged in. We therefore need to find where Ubuntu has decided to mount the SD card. It is easiest to switch to the directory '/media':

```
cd /media
```

and then type

```
ls
```

(for 'list') to display a list of files and directories within this directory. If there is more than one subdirectory, you can type

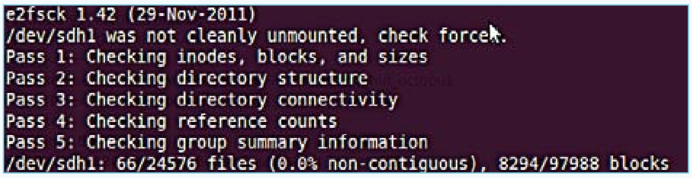
```
cd directory-name
```

to switch to a given directory, look inside using `ls`, and then `cd ..` to move back up one directory level into '/media'. The name of the directory where the SD card is mounted will typically consist of a long string of digits. The directory will contain the complete file system of the Linux board, including files with names such as 'zImage' and 'swapfile1'.

If you get lost in the file system, you can always return to your home directory using the command `cd` without any arguments. Another

First aid for a sick SD card

If a system that has been booted from an SD card is not powered down properly, using the `halt` or `poweroff` commands, it is possible that the file system on the card will be corrupted. This typically results in 'EXT2-fs errors':




```
Filesystem “EXT2-fs (mmcblk0p1): error: ext2_lookup:
deleted inode referenced: 694962”:
```

Fortunately we can usually rescue the file system using a Linux PC from the console.

First put the SD card into the card reader and use `dmesg` to determine what name it has been assigned. For example, if you see the following

```
[ 1549.424156] sd 7:0:0:2: [sdh] Assuming drive cache: write through
[ 1549.425624] sdh: sdh1 sdh2
[ 1549.427527] sd 7:0:0:2: [sdh] Assuming drive cache: write through
[ 1549.427533] sd 7:0:0:2: [sdh] Attached SCSI removable disk
[ 1549.730223] EXT2-fs (sdh1): warning: mounting unchecked fs, running e2fsck is recommended
```

it means that the first partition, which is the one we are interested in, has been given the name 'sdh1'. This contains a file system in ext2 format, which it is possible to repair. We have to unmount the file system

```
umount /dev/sdh1
```

before we can use the tool `e2fsck` (or equivalently `fsck.ext2`) to attempt the repair:

```
sudo e2fsck /dev/sdh1
```

The result should be as shown in the screenshot. From time to time the program will ask if certain actions should be carried out: you should normally answer 'y'. The result should be an error-free SD card!

handy command is `pwd`, which will tell you the path to your current directory.

Hello world!

To test the 'hello world' program, switch to the directory where the SD card is mounted and copy the file across:

```
cp ~/hello ./
```

Then we have to unmount the directory manually so that the operating system is forced to finish writing all the data to the card.

```
cd
sudo umount /media/directory-name
```

Here again 'directory-name' should be replaced by the name of the directory that the operating system chose when the SD card was mounted.

We can now move the SD card back to the Linux board and start it up. Connect to the board using a terminal emulator on the PC, as described in the previous instalment in this series [5].

Now, on the board, we switch to the top-level directory in the file system using the terminal emulator:

```
cd /
```

and run the program:

```
./hello
```

The result should be that 'Hello World!' appears in the terminal win-

dow (see **Figure 11**).

If the system is booted from the SD card, it is important to shut the system down in an orderly fashion when you have finished. For this we need the command

```
halt
```

It is then necessary to wait until the message `System halted` appears before it is safe to remove power: otherwise it is possible that not all files will be updated properly on the SD card. This can in turn result in 'EXT2-fs errors'. It is normally possible to recover the situation using a Linux PC: see the text box.

Bootloader and kernel

We shall now look at how we can build the two main components of the operating system, the bootloader and the kernel.
The source code we need (290 MB for the current version) is available from the Elektor website [3]. The simplest approach is to download the files using a browser: on a Linux machine the files will normally end up in the 'Downloads' directory.

When the download is complete, switch to the directory 'Downloads' and find the file called '120026-11.zip'. Move the file into your home directory using the command

```
mv ~/Downloads/120026-11.zip ~/
```

To unpack the file, switch back to your home directory using `cd` and enter the command

```
unzip 120026-11.zip
```

Figure 12 shows what you should see on the console.

Building the bootloader

The bootloader is a program which is copied from the SD card to the internal SRAM of the LPC3131 on system reset (assuming the jumpers are set correctly: see [5]). The following sequence of commands shows how we can compile the bootloader for ourselves and copy it to the SD card, to allow us to boot from the card. Before starting we need to install a couple of packages on the Ubuntu system:

```
sudo apt-get install patch libncurses5-dev
```

Now switch to the source code directory

```
cd  ElektorLinuxBoardDownload_20120509
```

unpack the tar file that contains the bootloader

```
tar xvzf bootloader.tar.gz
```

Figure 11. 'Hello world' running on the board.

then switch to the new 'bootloader' directory

```
cd bootloader
```

and unpack the source code proper:

```
tar xvzf apex-1.6.8.tar.gz
```

Now comes an important step if we want to be able to distribute any changes we make to the bootloader. We create a so-called 'working copy' of the source code and make changes only on that copy. Later it will be easy to create and publish a 'patch' that represents the changes we have made.

```
mv apex-1.6.8 work_1.6.8
cd work_1.6.8
```

In the source code tree we need to apply a couple of patches that are required to make the bootloader work with the Elektor Linux board.

```
patch -p1 < ../apex-1.6.8_lpc313x.patch
patch -p1 < ../gnublin-apex-1.6.8.patch
```

The build process for the bootloader is controlled by what is called a 'configuration file'. We have to copy the master version of this file into our directory, with the new name '.config'. Note that here the full stop before the word 'config' is very important: it marks the file as 'hidden' to the operating system.

```
cp ../gnublin-apex-1.6.8.config .config
```

At this point, if you installed the toolchain manually and did not arrange for the environment variables to be set in your .bashrc file, you will need to set them by running the script `. ~/set.sh`.

The build process can now be initiated:

```
make apex.bin
```

Normally at this point we would need to copy the bootloader firmware into flash memory on the microcontroller using a suitable programmer. Here, however, we can copy the firmware to the SD card using ordinary Linux commands.
With the SD card once again in the PC's card reader, run the

command

```
dmesg
```

to see the messages output by the kernel running on the PC. The results obtained on the author's PC are shown in **Figure 13**. As you can see, the SD card has been recognised as '/dev/sdh' and contains two partitions, called '/dev/sdh1' and '/dev/sdh2'. For safety it is best to unmount these partitions, as we will be copying the bootloader across by accessing the blocks on the SD card directly rather than via a file system:

```
sudo umount /dev/sdh1
sudo umount /dev/sdh2
```

The following command (where the 'sdh2' will need to be modified to reflect the results from the `dmesg` command above) will copy the file 'apex.bin' to the SD card in such a way that the LPC3131 can find it when booting:

```
sudo dd if=src/arch-arm/rom/apex.bin of=/dev/sdh2
bs=512
```

The command copies the bootloader code to the beginning of the second partition of '/dev/sdh' using a block size of 512 bytes. The Linux operating system does not guarantee exactly when the new blocks of data are actually written to the NAND storage on the SD card. The command

```
sync
```

is therefore needed to force the operating system to ensure that all pending blocks are written out and that the new bootloader code is safely stored on the SD card. We can now try to boot the Linux board from the SD card.

Building the kernel

Building the kernel is a similar process to building the bootloader. First we switch to the home directory using the command `cd`, and then into the source code directory.

```
cd  ElektorLinuxBoardDownload_20120509
```

We unpack the kernel source code

```
tar xvzf linux-2.6.33-lpc313x-gnublin-032012.tar.gz
```

switch to the kernel source directory

```
cd linux-2.6.33-lpc3131x
```

and start the build process that will result in a bootable kernel.

```
make zImage
```

We also need to compile the loadable kernel modules. The is simply done using the command:

```
make modules
```

With everything built we need to copy the kernel and the modules to the SD card. In our case the kernel 'zImage' happens to be there already, but it is worth practising the process for replacing the kernel. We need to copy the file 'arch/arm/boot/zImage' in the kernel source tree directly to the first partition on the SD card: the procedure is much the same as for the bootloader above, with the appropriate changes. Don't forget to unmount the partition!

A look at the possibilities

Now we are in the happy position of being able to rebuild the bootloader and kernel at will, we can look at the possibilities for making modifications to them, in particular to the kernel. Of course, we only have space here to look at this in general terms. You can take a first look at the Linux kernel configuration by typing

```
make menuconfig
```

from the directory 'linux-2.6.33-lpc3131x'. The result is shown in **Figure 14**. If, for example, you want to use a particular USB device with the Linux board, you have to enable to corresponding driver here.
You can navigate around the blue window using the arrow keys. The 'Enter' key opens and closes the menu. With a bit of hunting around you will be able to find drivers for various devices you recognise. In the next article in this series we will go into this subject in more detail.

Restoring the boot image

Since we are now beginning to get down to the nitty-gritty of how the *Elektor* Linux board works, it is a good idea to make an exact copy of the SD card for backup purposes.
Put the card in the PC's card reader and then check, using `dmesg`, what device name the operating system has chosen for it.
Again, for safety, unmount the partitions:

```
umount /dev/sdletter1
umount /dev/sdletter2
```

(where you should insert the appropriate character for 'letter', for example giving '/dev/sdb1' and '/dev/sdb2' or '/dev/sdh1' and '/dev/sdh2').

You can now take an exact copy of the card's contents:

```
sudo dd if=/dev/sdletter of=Image_SD_card_backup_
copy.img
```

This will take some time. You should now find a file in your current directory whose size is exactly equal to the capacity of the SD card:

```
ls -lh
```

You now need to put another SD card, exactly the same size as the original, in the PC's card reader. Again, find out the device name using `dmesg`. The following command will copy the card image you have just created onto the new card:

```
    sudo dd if=Image_SD_card_backup_copy.img of=/dev/
sdletter
```

This will take even longer than reading the original card. Assuming the command is successful you can now issue the `sync` command to ensure all blocks are actually written out to the card, and then try the new card in the *Elektor* Linux board. There is no need to `umount` the card as the file system on the card was never mounted in the first place: we wrote blocks directly to the card.

The above process is fairly straightforward, but unfortunately it does not work if the sizes of the two cards are not identical. Also, we sometimes want to create a new card with different partitions or with a different file system from those of the original card. These cases are handled by a graphical installer, which we shall look at briefly in the next article in this series.

What the future holds

In this instalment we have made good progress on the route to understanding our embedded GNU/Linux system. We have a development environment in place, we can now build our own bootloader and kernel, and we have compiled and run a small program. In the next article we will take a quick look at the structure of the source code for Linux so that we can be in a position to write our own driver for a particular piece of hardware. We will also see how easy it is to write programs using scripting languages.

(120180)

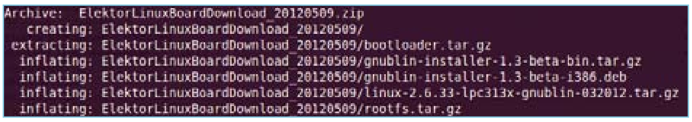


Figure 12. Messages that appear when unpacking the software download.

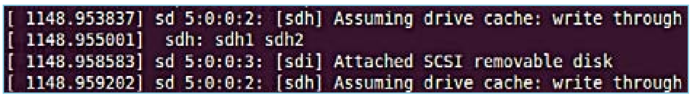


Figure 13. The messages from the kernel show that the SD card has been assigned to '/dev/sdh'.

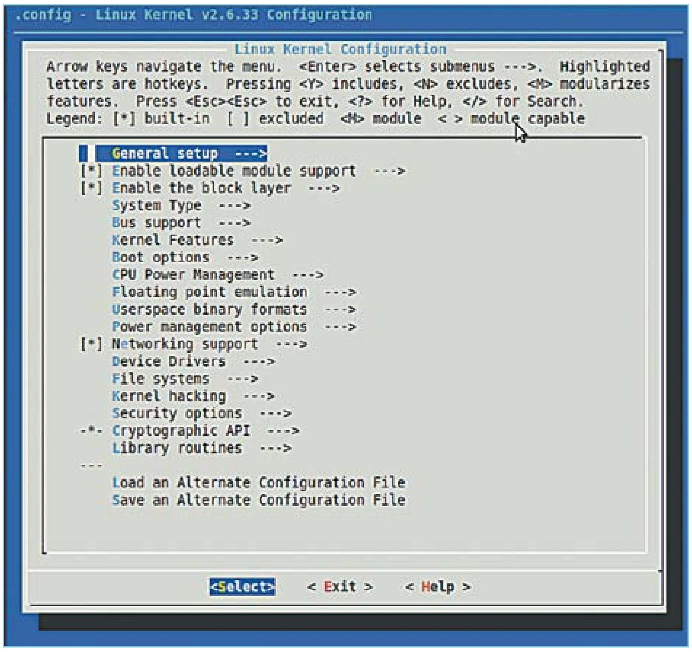


Figure 14. Configuring the Linux kernel.

Internet links

[1] sauter@embedded-projects.net

[2] http://www.ubuntu.com

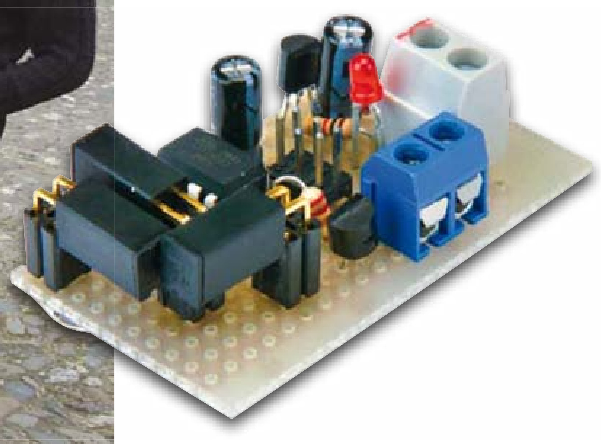
[3] http://www.elektor.com/120180

[4] http://www.virtualbox.org

[5] http://www.elektor.com/120146

Motorbike Alarm

electronic sentry



By Pierre Rigaill (France)

The principle is a very simple one: once the bike has been set on its stand, three carefully-positioned mercury switches give position information to a microcontroller, which it then memorizes when it is powered up. The surveillance starts right away; any change in the bike's position will upset the configuration of the three switches and set off the alarm.

My circuit fits into a tiny case, barely bigger than a matchbox.
I've already fitted this alarm to two bikes, a Kawasaki 650 and an Aprilia SX50. Elektor in turn has fitted it to a vintage Honda Goldwing GL1200 that you can see in the photos. For environmental reasons, the Elektor lab has preferred to replace my mercury switches with roll-ball tilt switches, cheap and less polluting.

When power is applied, a very brief audible signal indicates the start of a 30 s delay during which you can still move the bike; once this delay has elapsed, the bike's position is memorized. This is indicated by a second, very brief audible signal. As soon as the microcontroller detects a change in the position of the switches, after a final delay of 30 s (to allow for accidental jolts), the alarm is set off for 30 s. If at the end of this 30 s period the bike is still not in its original position, the alarm will continue to sound.

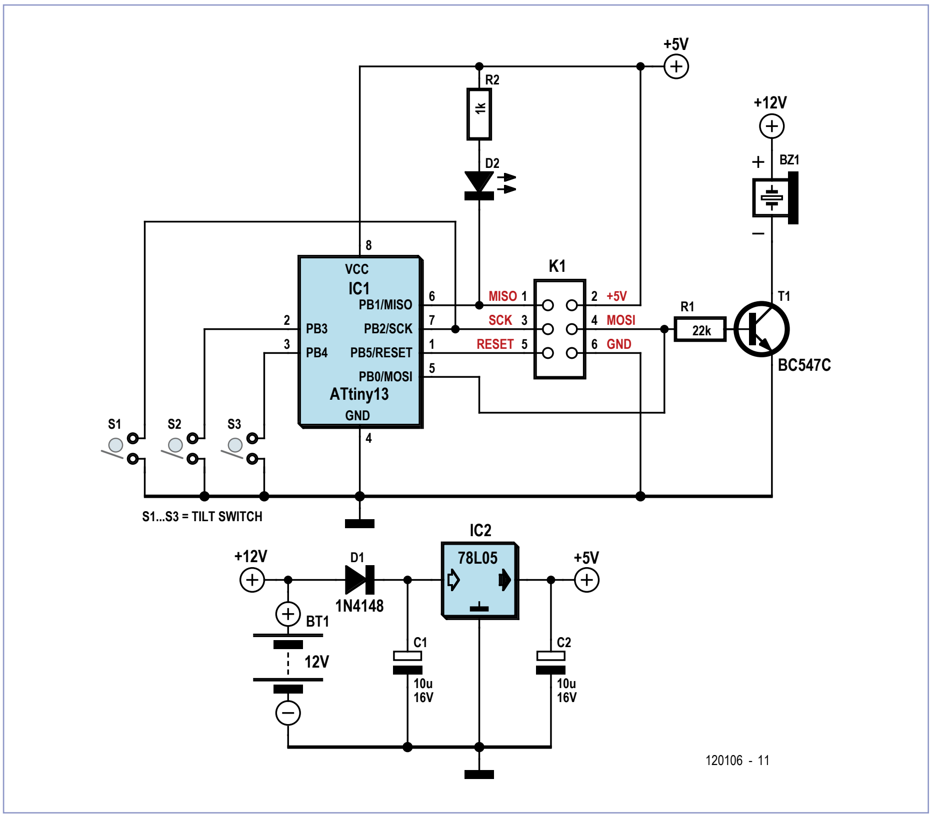
The µC in my prototype is an ATtiny13 in a DIP package, mounted on prototyping board. To program it, I used a small connector from TE CONNECTIVITY / AMP, part number 7-215079-6, which is smaller than an HE10. I also fitted the same connectors to my AVR-ISP probes. Their pinout follows the Atmel standard for the AVR family. If you really want to go for miniaturization, you can also program the microcontroller then solder it in directly, which saves some space.

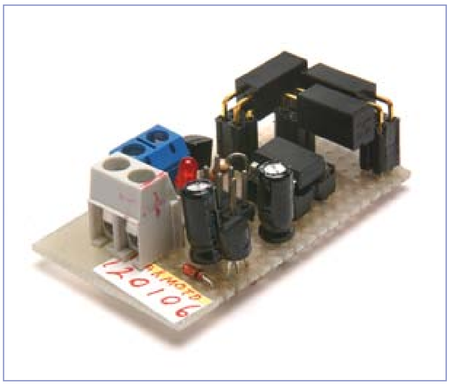
As shown in the photo, two of the three 90° switches (S1 to S3) are arranged head-to-tail in a sideways direction, and slightly inclined in opposing directions, so as to obtain reliable detection of any movement of the bike when it is up on its side stand.

The third switch is arranged longitudinally, at right-angles to the other two; this one will be more likely to detect the jolts caused by taking the bike down off a main stand. The three switches that form the tilt detector are connected to ports PB2, PB3, and PB4 on the µC. To prevent these inputs from floating, their internal pull-ups are enabled in the ATtiny13.

The switches are fitted in sockets so they can be easily removed while programming the µC, in order to avoid any possible conflict between internal and external logic levels; in any event, the switch connected to PB2 must be open during programming, otherwise the SCK line will be at 0 and programming will be impossible.

If you want to, port PB1 will let you connect





an LED (D2), which will flash to show the alarm is in service.

The PB0 output drives the 'siren' via T1. The sounder is not a passive piezo resonator, but an active type with built-in oscillator, powered from 12 V, which emits a piercing sound. When it is operating, it consumes around 150 mA.

In stand-by, the consumption of the alarm is of the order of 1.5 mA. As the circuit in its present state does not have a self-contained power supply, the bike power needs to be present during stand-by.

You'll need to find a place to hide the alarm unit that is discreet and relatively difficult to get at. The aim of my system is first and

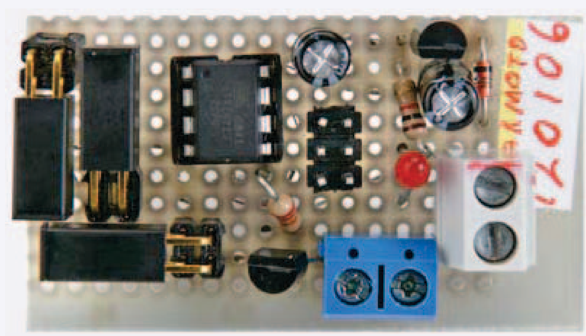
COMPONENT LIST

Resistors
R1 = 22kΩ
R2 = 1kΩ

Capacitors
C1,C2 = 10µF 16V, radial

Semiconductors
D1 = 1N4148
T1 = BC547C
IC1 = ATTINY13-20PU (Atmel), DIP-8, Elektor # 120106-41
IC2 = 78L05

Miscellaneous
S1,S2,S3 = tilt switch, 90 degrees, e.g. RBS04020



foremost dissuasion, and I'm well aware of the fact that on certain models of bike, like the SX50, a determined thief will very quickly lift the saddle and cut the battery cables!
To conclude, all that remains is to find a discreet location to hide the reset switch (to be connected between pins 5 and 6 on K1) that will enable the protected bike's legitimate owner to disable the alarm.

Software

The very simple program [1] has been written in C using the CodeVisionAVR tool (published by HP Infotech) and consists of just a single C file (main.c). Once compiled, it occupies less than 50 % of the Tiny13's flash memory. In order to minimize power consumption, the Tiny13 uses its own internal 128 kHz oscillator. For what it has to do, there's no point clocking it like mad at 8 MHz! The description of the FUSE BITS is given as a comment within the *main.c* file.

At the start of the file, we find the type definitions (*typedef*), a few *defines*, and the global variables, whose names I have made always start with G_ in order to differentiate them from the local variables. The global variables are the ones that are shared between the background task and the *timer 0* interrupt. The latter has been configured in CTC mode so as to generate an interrupt every 100 ms. This 100 ms timebase is used both for the countdown for the 'anti-jolt' delay and for handling the flashing of the LED. This gives a 100 ms flash every 500 ms during the power-on delay period, then one flash every 5 s in stand-by mode.

Divided by 10, this 100 ms timebase is also used to obtain seconds and handle the delays that use an increment of 1 s.

The `interrupt [TIM0_COMPA] void timer0_compa_isr(void)` function is the *timer 0* interrupt routine, and like any interrupt function, neither receives any parameters nor returns anything. It makes use of two static variables that are retained from one interrupt to the next.
We come next to the `void main(void)` function. Naturally, this is the main program, which cannot receive any parameters nor return any values either!

The *main* uses a single variable *memo_sw* for memorizing the state of the switches.
Here we find the initialization of the microcontroller, the interrupt enabling, the power-on beep, and then the infinite loop that makes up the background task. It's here that we test the delays being counted down in the *timer 0* interrupt and verify the state of the switches. My comments [1] will help you understand how it works.

(120106)

Internet Links

[1] www.elektor.com/120106

Source code excerpt; full: program available for free downloading [1]

```
/*---------------------------------------------------
  I Fonction     : main                              I
  ---------------------------------------------------
  I Action       : main program                      I
  I In Param     : nada                              I
  I Return       : nada                              I
  I                                                   I
  --------------------------------------------------*/
void main(void)
{
   byte memo_sw;                    // tilt status check

   // Crystal Oscillator division factor: 1
   #pragma optsize-
   CLKPR=0x80;
   CLKPR=0x00;
   #ifdef _OPTIMIZE_SIZE_
   #pragma optsize+
   #endif

   // Input/Output Ports initialization
   // Port B initialization
   // Func5=In Func4=In Func3=In Func2=In Func1=In Func0=Out
   // State5=P State4=P State3=P State2=P State1=P State0=0
   PORTB=0x3E;
   DDRB=0x01;

   // Timer/Counter 0 initialization
   // Clock source: System Clock
   // Clock value: 0,500 kHz
   // Mode: CTC top=OCR0A
   // OC0A output: Disconnected
   // OC0B output: Disconnected
   TCCR0A=0x02;
   TCCR0B=0x04;
   TCNT0=0x00;
   OCR0A=0x31;
   OCR0B=0x00;
```

‘Spoon’ soldering

By Thijs Beckers (Elektor Editorial & Labs)

Soldering SMD ICs seems to scare off many inexperienced electronics fans, despite efforts by ‘old hands’ and other experts to lift the jinx. “It’s just too small”, “I don’t have the right tools”, “They’re all in mum’s hoover” et cetera. I refute those arguments. Here’s another ‘trick’ to solder small pitch ICs.
As you can see in the photographs, this particular IC doesn’t come in the smallest of packages — it’s an SN20086APF chip from Sonix in an 48-pin LQFP package on a rather ancient memory stick with 128 MB of storage. However with its 0.5 mm pitch (distance between pins) it’s perfect for demonstrating another way of soldering this type of component.
Here, I am hoping to introduce you to a method of soldering an LFQP IC using a relatively large solder tip. First of all, make sure the IC is positioned correctly and secured in its place, for example by soldering two opposite pins. Next, apply a little solder flux along the pins. Now use a chisel type solder tip — a ‘spoon’ tip — with a little solder in its ‘bowl’ and slowly move the tip along the pins to be soldered — bowl up, and angled at about 30 degrees, see the photographs. Enjoy watching the surplus solder in the bowl at the tip flowing to the pins and solder pads, joining them ever so neatly.
Sure, some practicing will be needed, but it’s not too difficult. After a few try-outs you quickly arrive at the right speed of movement and amount of solder to apply and you’ll be off looking for more small ICs waiting to be by secured on a board ‘spoon-feed’ soldering.
Ah, you’re not into SMT soldering at all and think 0.5 mm pitch is too ambitious as a first attempt? Try our simple electrosmog detector kit, the TAPIR, with a few SMDs only. It’s a fun project with lots of documentation available on line: www.elektor.com/120354.

(120234)

USB: current unlimited!?

By Raymond Vermeulen (Elektor Labs)

After working on a number of USB-related projects it occurred to me something odd was happening with the power supplied to a connected USB device. I started to doubt the consensus that a USB host device limits the supplied maximum current to 100 mA and only after a connected device identifies itself as being a high-power device, this limit would be upped to 500 mA.
Using the recently published USB power meter (‘I’ve got the USB Power’, July & August 2012) and a simple dummy load consisting of two 22 ohm/10 watt resistors in parallel I ran some measurements on several USB connections available on PCs in the Elektor Labs. All measurements led to the same conclusion: apparently there is no power limiting — at least not to 100 mA!
Let’s assume that no current limit is applied in the case of USB connections not featuring data communication — after all, resistors aren’t terribly ‘communicative’. To disprove the assumption, I connected a device (project with an ATmega32U4, to be published soon) that negotiates specifically being a low power device, needing only 100 mA and connecting a load (resistor) in parallel on the USB bus power lines. Again, the full 500 mA were still available!?
From these observations I can only conclude that modern PCs and laptops always offer the full 500 mA current capacity on their USB buses. I suspect the 100 mA limit originates from the early USB days in the nineties, and that recently there is no reason for limiting the maximum available current anymore, possibly due to the general availability of more potent components.
If anyone has information on this or a better explanation of the phenomenon, please write to us (preferred e-mail: r.vermeulen@elektor.nl) so we can enlighten ourselves and the community with unlimited knowledge.

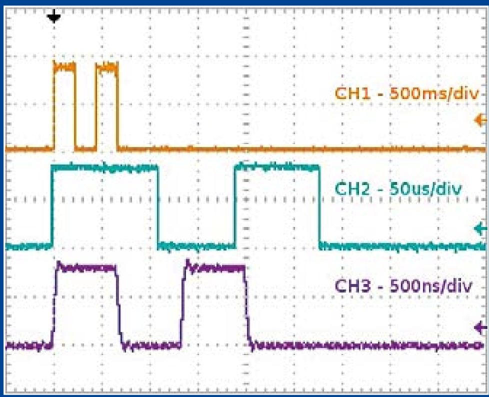
(120436)

GPIO access on Elektor Linux board

By François-Xavier Maurille (Elektor Labs Intern)

At some point during the development of a musical project based on the Elektor Linux board I found that an SPI DAC was called for. It turned out I needed to work with 24-bit words, but the SPI hardware interface inside the chip wasn't capable of processing data in that format.

Consequently the chip select (CS) output had to be realised with a GPIO pin, which proved to be tough to implement in the operating system: the response time after a command was given was far from insignificant due to the complex and busy exchange between the application and the hardware.



The illustration shows the results of my attempts of speeding up the CS sequence '1010'. In my first attempts I used the Linux file system and its *fprintf C* function. CH2 shows the results of the command `fprintf("/sys/class/gpio/gpio11/value", "%d", state).` A minimum response time of 90 µs was feasible, which was far too slow for my application.

Next I tried to access the GPIO pin using the *echo system* command. CH1 shows the output signal after the command `system("echo 1 > /sys/class/gpio/gpio11/value")` was given. This didn't speed things up either. In fact, it was even slower than my first attempt: about 200 ms elapsed before the CS output changed state.

Then I tried to access the microcontroller's GPIO registers directly. This is much more difficult than using the file system, but it sure proved a lot faster. Thanks to guidance on http://forum.gnublin.org I was able to implement a few functions to set and reset GPIO pins using a pointer and offsets on the IOCONFIG register address (given by *mmap()*) to access the MODEx registers. Signal CH3 describes the output after the command `*(unsigned int *)(ptr + GPIO_OFFSET + GPIO_MODE0) = 1 << nGPIO;`. This command resulted in a response time of (about) 700 ns, which was more to my liking.

Some observations:

- Pay attention: The **4th** MODEx bit is used for GPIO**4**, while the **5th** MODEx bit defines GPIO**11** (CS) as set/reset pin (and not GPIO**5**).
- With your Linux Board you'll be able to 'work in the future', when the internal date of the board is not in sync with your PC's internal date. I received the following warning while working with the board: *"make: Warning: File 'GPIO.c' has modification time 52 s in the future"*.

(120457)

SD Card Correction Script

By Francois-Xavier Maurille (Elektor Labs Intern)

The popular Elektor Linux Board (article series started in the May 2012 edition) provides an easy way for starters to engage in a Linux environment. Albeit the hardware is working fine, there might be a complication with the software. Those of you already using the Elektor Linux board may have experienced the odd problem with an SD Card appearing to be corrupted. The notice "`EXT2-fs (mmcblk0p1): error: ext2_lookup: deleted inode referenced: 694962`" might pop up.

An easy instruction is given in this month's edition explaining how to fix this error. But if you're too lazy to copy-paste the instructions, or if you are not sure how to proceed, you may download a little piece of software that does it all for you. It's a small bash script I wrote, meant to be run on a Linux PC. It runs exactly the same commands as described in the article inset. With one exception: they are run automatically. The script looks for the corrupted device name using the `grep` and `sed` commands, unmounts the device and runs the `e2fsck` command.

This script is able to correct your SD Card in a very straightforward manner. You just need to follow the on-screen instructions, and answering by pressing 'Enter' or 'y'. These are the concise instructions:

After downloading the bash script from [1], browse to the directory where you stored the file, and unpack it (right-click on it and select *Extract Here*). Before the first run, the script has to be made executable. Start a terminal session, browse to the script's directory and enter the following line in the terminal window: *sudo chmod 777 correctSD.sh*. Now you should be able to execute the bash script from within the terminal (enter *./correctSD.sh*) and let it take care of the corrupted SD card. A small additional README text file provides some extra information on the correct procedure.

This is of course *not* an excuse to press the RESET button on the board all the time...

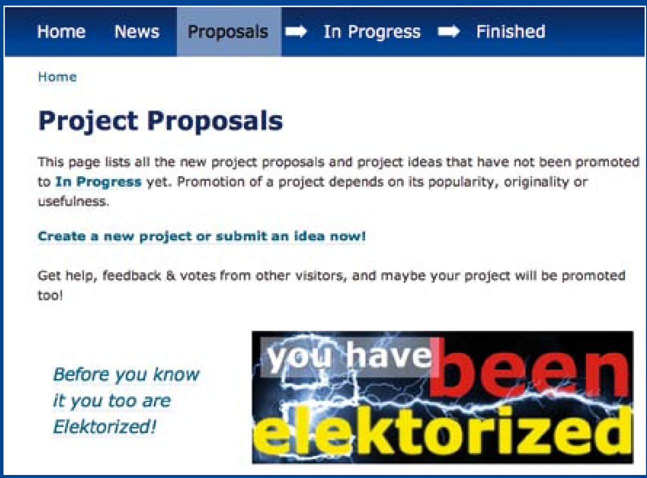
(120443)

Internet Link

[1] www.elektor.com/120026

E-LABS INSIDE

Elektor-projects.com 4U2



Home News Proposals ➡ In Progress ➡ Finished

Home

Project Proposals

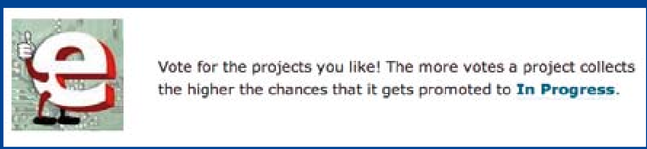
This page lists all the new project proposals and project ideas that have not been promoted to **In Progress** yet. Promotion of a project depends on its popularity, originality or usefulness.

Create a new project or submit an idea now!

Get help, feedback & votes from other visitors, and maybe your project will be promoted too!

Before you know it you too are Elektorized!

you have been elektorized



Vote for the projects you like! The more votes a project collects the higher the chances that it gets promoted to **In Progress**.

Elektor Labs has been expanded and now **you too** can get involved in the development of your favourite magazine's projects. In fact we encourage you to join the international Elektor community — show and discuss your own ideas and developments out there and see what Elektor Labs is working on. If your project is worth its salt it may get elected for developing into a real Elektor Project!

Fresh project ideas, circuit sketches and other e-doodling gets entered in the *Proposals* section. The most interesting or appealing ones are picked up by our lab team and developed into a project (and matching article) of the kind Elektor is famous for. And the fame is yours too.

Take a look at the *Switched 7905 Replacement* in the *In Progress* section. Publication of this project is due in the October 2012 edition of Elektor; on our Projects website you can get a sneak preview of the project and actually follow some of the designer's steps as he progresses with the design.

The section called *Finished* contains wrapped up projects, like Platino and the Improved Radiation Meter. Of course this doesn't mean the end of a project...

(120484)

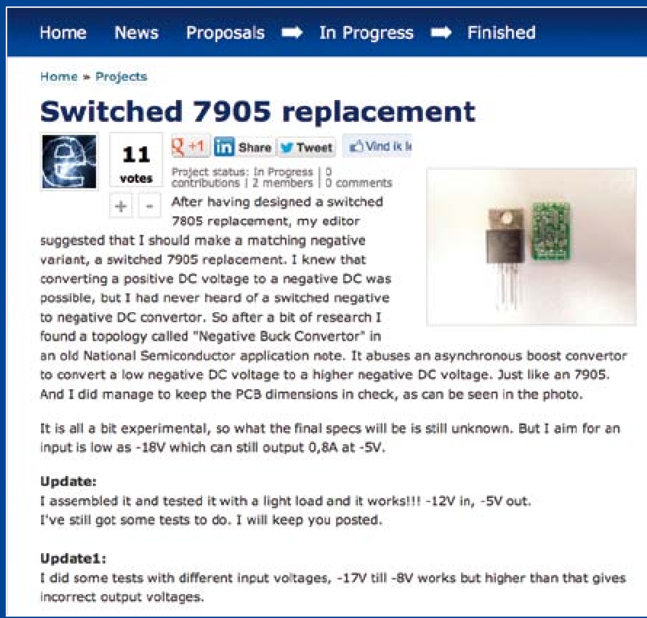
Please note: Read-only access to our Elektor Projects website is free for everyone, but only Elektor Plus members can actively participate. If you are an Elektor Plus member, use your Elektor Plus credentials to enter Elektor Projects. If you don't remember your password, you can request a reminder using the email address you registered for your Plus membership.

Home News Proposals ➡ In Progress ➡ Finished

Home

Projects in Progress

This page lists all projects currently under active development and not yet (completely) finished. Projects on this list may get support from the Elektor team and even make it to print in **Elektor Magazine** and/or become a ready-made product in the shop. That's right, you can **make money from your project!**



Home News Proposals ➡ In Progress ➡ Finished

Home » Projects

Switched 7905 replacement

11 votes

+1 Share Tweet Vind ik leuk

Project status: In Progress | 0 contributions | 2 members | 0 comments

After having designed a switched 7805 replacement, my editor suggested that I should make a matching negative variant, a switched 7905 replacement. I knew that converting a positive DC voltage to a negative DC was possible, but I had never heard of a switched negative to negative DC convertor. So after a bit of research I found a topology called "Negative Buck Convertor" in an old National Semiconductor application note. It abuses an asynchronous boost convertor to convert a low negative DC voltage to a higher negative DC voltage. Just like an 7905. And I did manage to keep the PCB dimensions in check, as can be seen in the photo.

It is all a bit experimental, so what the final specs will be is still unknown. But I aim for an input is low as -18V which can still output 0,8A at -5V.

Update:
I assembled it and tested it with a light load and it works!!! -12V in, -5V out.
I've still got some tests to do. I will keep you posted.

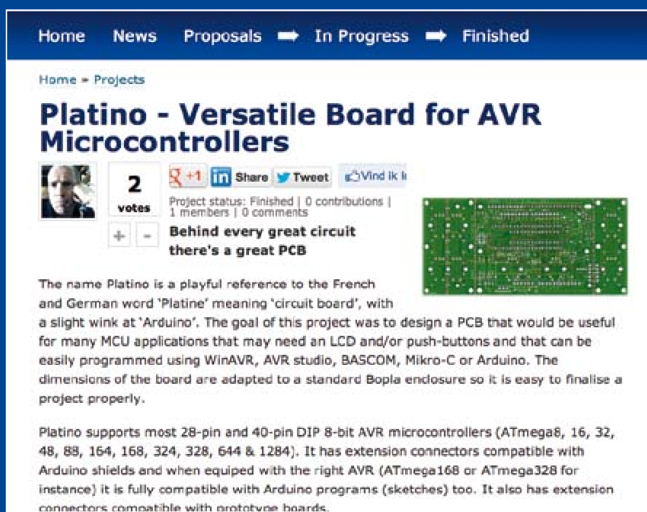
Update1:
I did some tests with different input voltages, -17V till -8V works but higher than that gives incorrect output voltages.

Home News Proposals ➡ In Progress ➡ Finished

Home

Finished Projects

The projects on this page are no longer actively developed because they are either finished or have reached a dead-end. Finished projects may still be updated on occasions when a bug is found, a component has turned obsolete or an improvement was added. Some of these projects have been published in **Elektor Magazine**, some have products for sale in the online **Elektor Shop**.



Home News Proposals ➡ In Progress ➡ Finished

Home » Projects

Platino - Versatile Board for AVR Microcontrollers

2 votes

+1 Share Tweet Vind ik leuk

Project status: Finished | 0 contributions | 1 members | 0 comments

Behind every great circuit there's a great PCB

The name Platino is a playful reference to the French and German word 'Platine' meaning 'circuit board', with a slight wink at 'Arduino'. The goal of this project was to design a PCB that would be useful for many MCU applications that may need an LCD and/or push-buttons and that can be easily programmed using WinAVR, AVR studio, BASCOM, Mikro-C or Arduino. The dimensions of the board are adapted to a standard Bopla enclosure so it is easy to finalise a project properly.

Platino supports most 28-pin and 40-pin DIP 8-bit AVR microcontrollers (ATmega8, 16, 32, 48, 88, 164, 168, 324, 328, 644 & 1284). It has extension connectors compatible with Arduino shields and when equiped with the right AVR (ATmega168 or ATmega328 for instance) it is fully compatible with Arduino programs (sketches) too. It also has extension connectors compatible with prototype boards.

Plug-o-(d)rama 2.0

By Thijs Beckers (Elektor Editorial & Labs)

In this year's March edition you got a glimpse of an unsystematic selection of old connectors from Managing Editor Jan Buiting's private collection of vintage electronics. The page triggered numerous responses from our readers, which were gratefully appreciated. Many readers managed to provide descriptions and images of their oldest and most peculiar connectors, many of which we think are worth showing. So here's a random selection of pictures received, with their descriptions.

The first collection shows Dr. Klaus Rohwer's set of Hirschmann plugs, some of which probably date back to the 1950s. The first four are power plugs, the big one is designated as 'UPO' — Unidentified Plug Object. We suspect it's for a telex machine; one of my fellow lab workers thinks he's seen something similar on an old telephone switchboard in a former military command centre.

This second set of connectors shows various audio plugs once used at the *Westdeutscher Rundfunk Köln*. Elektor reader W. Richter used to play around with these starting from the 1950s. Various adapter plugs were also constructed to be able to link different connector plugs. Says Mr. Richter: "the only adapter we didn't construct was from Lemosa to Gardena."

This plug and socket belong to a 'Pansanitor' machine, which dates back to 1928 (see also this month's Retronics article). According to proud owner, Mr. Butte from Germany, these original connectors are made of glazed porcelain. They are probably the oldest plugs of which we have received photographs so far.

This little collection was pictured by W. Haas. We start with an 'LMK' radio antenna connector in the upper left corner, which has one horizontal and one vertical oriented prong. Next to it is a 240-ohm VHF antenna connector for balanced-feeder cable. Then two white symmetrical VHF Band III 240-ohm plugs with horizontal prongs. Next to those are two red, symmetrical 240-ohm connectors for UHF band IV/V with vertical prongs. The bottom right connector is a plug with cubed prongs, suitable for VHF and UHF. The rest of the connectors pictured are various adapter plugs, from banana to single pin; the blue one suited with a wire insert and fastening screw.

P. van de Meerendonk from The Netherlands sent us, among others, this picture of another 'UPO'. It looks like a tube socket, but it's not. As far as we could see, it is some sort of octal-style plug made from Bakelite. Application unknown.

We received many more photographs and even some descriptions of older connectors (without photographs). We even triggered some remarks on our descriptions. General consensus on connector #10 in the original Plug-o-(d)rama article is a balanced 240-ohm antenna connector for radio, suitable for 3 and 4 mm holes; 3 mm supposedly used for VHF and UHF frequencies, 4 mm for VHF FM radio.

Connector #11 would be the chronological successor of #10. This polarized connector offers connections for an AM 'antenna' (vertical) and 'ground' (horizontal).

Suggestions came in that plugs #6 weren't used for loudspeaker connections unless they where the Bakelite version; high impedance loudspeakers were often directly connected to the anode, which could lead to dangerously high voltages on the cables and connectors.

(120303)

Low-cost 60 MHz Sig Gen

With PWM mode and incremental control

By Peter de Bruijn (Netherlands)

This low-cost signal generator has an impressive frequency range from 250 Hz to 60 MHz. It can generate PWM signals from 250 Hz to 60 kHz, which are suitable for testing devices such as LED dimmers, ultrasonic transducers, voltage converters and drive systems for electric bicycle motors. It can also be operated in oscillator mode with a frequency range of 1 kHz to 60 MHz. This is useful for applications such as testing a DCF aerial at 77.5 kHz, generating an adjustable 20 MHz clock signal for a processor, or testing resonant circuits. The signal generator is controlled by four capacitive touch keys.

enclosure equipped with a well defined front panel. The CPS sensors can be made from small pieces of copper PCB material measuring 1 × 1 cm, attached to the enclosure by hot-melt glue.

The source code and hex code for the PIC microcontroller can be downloaded from the web page for this project [1]. A brief user guide (120111-W.doc) is also available on the web page.

(120111-I)

Internet Link

[1] www.elektor.com/120111

low-cost generator with an impressive frequency range

Specifications

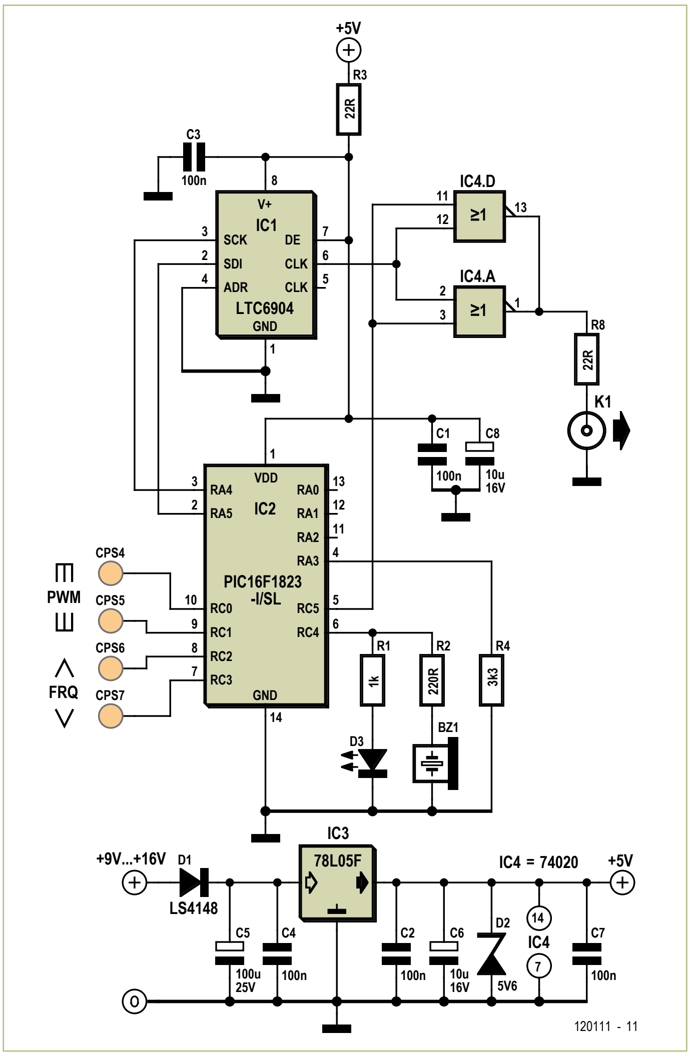
• Initial frequency:	10 kHz / 5 V	OSC mode
• Mode 1: PWM		
250 Hz to 60 kHz	Drift: 1%	Accuracy: 1%
• Mode 2: OSC		
1 kHz to 60 MHz	Drift: 0.05%	Accuracy: 0.75%

The main components of the circuit are a PIC microcontroller and a programmer oscillator module (type LTC6904). The oscillator module is controlled by pins RA4 and RA5 of the PIC device. The touch keys are connected to inputs RC0–RC3. The PIC device used here allows sensor keys to be connected directly to several inputs, eliminating the need for extra hardware. Output RC4 drives a piezoelectric beeper that generates a tone when a key is touched. LED1 also provides a visual indication. Two paralleled XOR gates are connected to the output of the LTC6904 to provide sufficient current at the signal generator output. PIC output RC5 is responsible for PWM signal generation.

IC3 provides a stabilised supply voltage. The combination of R3, C1 and C8 provides good supply decoupling. A 5.6 V Zener diode is included as extra protection for the LTC6904 in the event that an excessive voltage is accidentally applied to the output connector. The signal generator can be powered from a standard AC power adapter with an output voltage in the range of 9 to 16 V.

The PCB layout must provide screening between the capacitive touch sensors and the oscillator output. This can be achieved by placing a ground plane between the oscillator and the CPS IC.

The relatively few components can easily be assembled on a piece of prototyping board. The author fitted the circuit board in a tiny



Electronics for Starters (7)

Blinkers and oscillators

In the previous instalment we worked with static flip-flops and Schmitt triggers. Now things get a bit more dynamic, with capacitors providing the necessary feedback. You will see LEDs blinking and flashing against an audio backdrop provided by signal generators. After all, making things blink and beep is one of the main functions of electronics.

By Burkhard Kainka (Germany)

A bistable circuit that automatically and continually switches states without any outside influence is called a multivibrator or an astable multivibrator. **Figure 1** shows an example of an astable multivibrator circuit, and as you can see right away, in this circuit the feedback is provided capacitors. If you use electrolytic capacitors for this, you must pay attention to the polarity because the voltage on the associated collector is (on average) higher than the voltage on the opposite base. The circuit remains in a stable state only as long as it takes for the capacitors to be charged or discharged. After this the circuit flips to the opposite state and the process starts again.
A practical experiment with two 10 µF capacitors yielded a fairly low LED blinking rate with a period of around 1 second. You can adjust the toggle rate of the multivibrator over a wide range by changing the capacitor values. Experiments using relatively small capacitors and capacitors with differing values are also worthwhile. With a pair of capacitors having values of 100 µF and 100 nF, the circuit generates short flashes of light from one of the two LEDs. With a pair of 100 nF capacitors it produces rapidly flickering light.

Simplified multivibrator

The multivibrator circuit can be modified to operate with just one capacitor. The circuit basically needs two transistors operating in common-emitter mode, each acting as an inverter that changes the phase of its input signal by 180 degrees. These two stages can be directly coupled to eliminate one of the capacitors, as shown in **Figure 2**.
To ensure reliable oscillation, this circuit must be dimensioned to have an operating point in the middle of the characteristic range in the absence of feedback. Otherwise the output transistor would be either completely cut off or driven fully on. The overall circuit would therefore not have enough gain to start oscillating. In this example, strong negative feedback on the first transistor (provided by the 10 kΩ resistor between the collector and the base) yields an operating point in the middle of the range. However, the feedback via the RC network is stronger than the negative feedback, with the end result that the output transistor is alternately cut off and driven into saturation.

It's a good idea to first put together the circuit without the feedback capacitor. The LED should light up dimly because the output transistor is not fully switched on. With the capacitor fitted, the LED will alternately be full on or full off. With a 22 µF capacitor, the LED blinks approximately once per second. The circuit also works with smaller capacitors, down to a value of 10 nF. As you reduce the capacitor value, the blinking gradually changes to rapid flickering. If you connect an acoustic transducer to the output, you will hear a clacking sound.

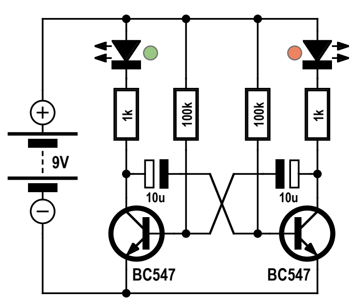


Figure 1. Multivibrator circuit.

LED voltage converter

Red LEDs need 1.5 to 2 V, while blue or white LEDs need as much as 3 to 4 V. This is usually handled by connecting three batteries in series to provide 4.5 V. A series resistor is then used to reduce the supply voltage to the operating voltage. It would be better to be able to manage with just 1.5 V, which means we need a voltage converter.
The key component here is a small fixed inductor rated at 1.5 mH. This component looks like resistor, but it has a small ferrite core and a wire coil under the protective lacquer. If you wish, you can also make your own inductor for this purpose. Around 200 turns on a ferrite rod will do the job.
The circuit shown in **Figure 3** is another simple multivibrator. The current through the inductor is switched on and off at a high rate. Here the inductor acts as a magnetic energy storage device. Each time it is switched off, it generates an induced voltage that adds to the battery voltage. The magnitude of this voltage depends on the connected load. It adapts automatically to the load, so a white LED receives a higher voltage than a red LED. Most voltage converters of this type

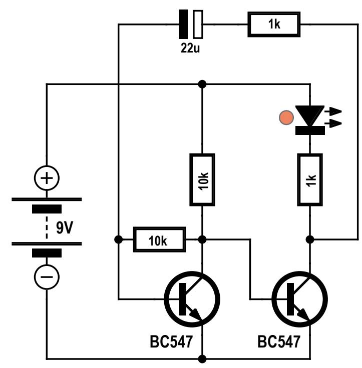


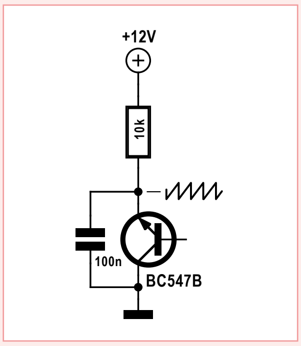
Figure 2. Simplified blinker circuit.

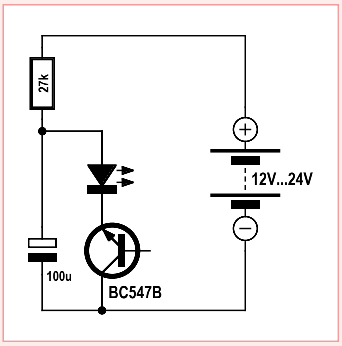
NPN sawtooth signal generator

A sawtooth generator can also be built with a single transistor operated in a highly unusual manner. Here the NPN transistor is wired the wrong way round in the circuit, with a positive voltage on the emitter, and the base lead is left open. The voltage on the capacitor gradually rises to approximately 9 V. At this point the transistor suddenly starts conducting and discharges the capacitor to approximately 7 V. The data sheet for the transistor doesn't say anything about this, and each transistor behaves somewhat differently. It's worth trying a variety of transistors in this circuit.

The discharge current pulse is strong enough to drive an LED. This requires a supply voltage greater than 12 V. The circuit works very nicely with a pair of nearly exhausted 9 V batteries. The LED keeps blinking for a long time as it sucks the batteries dry, with a gradually decreasing blink rate.

When operated this way with an 'inverted' voltage between the emitter and the collector, the transistor has a characteristic curve with a negative slope, which can easily be measured. The base-emitter junction exhibits the well-known avalanche breakdown effect at approximately 9 V. At this voltage the high electric field strength in the thin reverse-biased junction region causes the charge carriers to move so fast that they dislodge other charge carried from the crystal lattice. As result the number of charge carriers, and with it the current, rises very quickly. This is the same as what happens in a 9 V Zener diode, but a Zener diode has a positive internal resistance.





There's another factor involved with an inverted transistor. Here the emitter and collector switch roles, but due to the essentially symmetrical structure the transistor also operates in this inverted condition. The operating principle of a transistor is that some of the charge carriers that enter the base layer pass through the reverse-biased junction on the other side. In this case avalanche breakdown is occurring in the reverse-biased junction region, so there are even more charge carriers available to dislodge additional charge carries from the lattice. This results in a self-reinforcing avalanche effect.

also have a rectifier and a smoothing capacitor. Here they are not necessary because the LED acts as its own rectifier. A pulsating DC current flows through the LED. The average value of this current is somewhat less than the battery current because the voltage is higher. The overall efficiency of this circuit is higher than with the usual approach of using a higher battery voltage and a series resistor.

Audio generator

If we fit our simple multivibrator with a relatively small-value capacitor, it will generate a signal in the audio frequency range. A piezoelectric transducer (buzzer or beeper) connected to this circuit with produce an audible tone (**Figure 4**). A piezoelectric transducer has some of the characteristics of a capacitor and therefore affects the audio frequency. As a diversion, you can try a little experiment with this circuit. If you touch the transducer with your finger or with a hard object, the tone (frequency) and the sound level both change. The vibrating piezoelectric disc generates an AC voltage that affects the signal generator. To a lesser extent, acoustic echoes can also affect the signal generator.

The frequency can also be modified by adjusting the value of the resistor in the feedback path. You could use a potentiometer for this, or you could use a light-dependent resistor (LDR). In the latter case the audio tone depends on the amount of light striking the LDR.

With the circuit shown in **Figure 5**, you can distinguish not only the brightness of the light, but also different types of light. Rapidly fluctuating artificial light modulates the

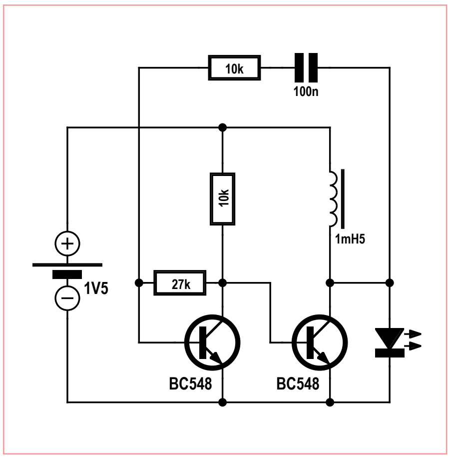


Figure 3. LED voltage converter.

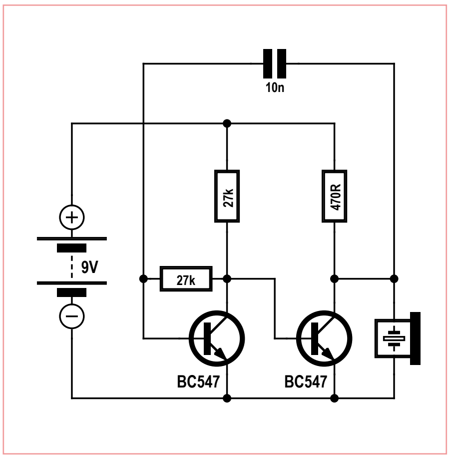


Figure 4. Driving a piezoelectric buzzer.

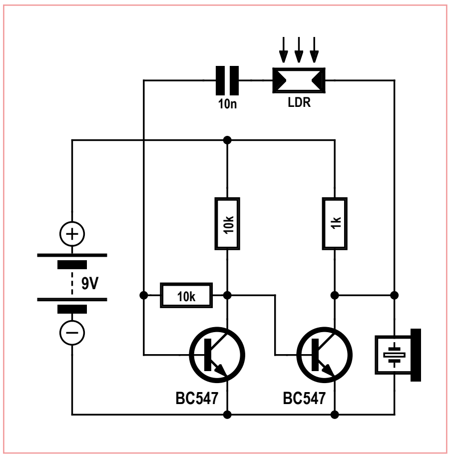


Figure 5. An adjustable audio signal generator.

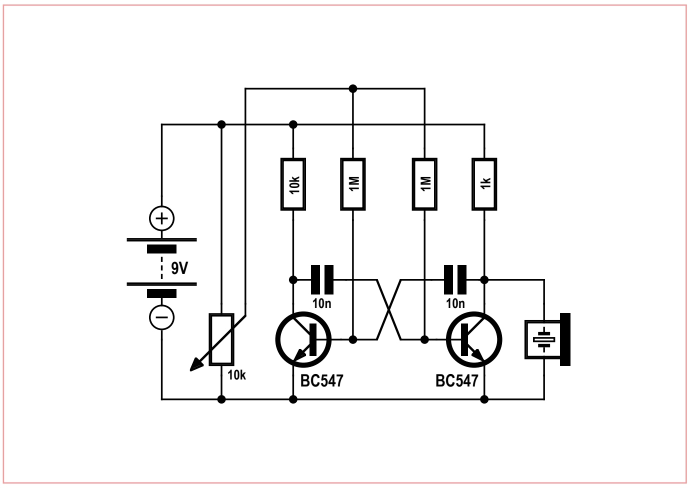


Figure 6. Using a multivibrator as a VCO.

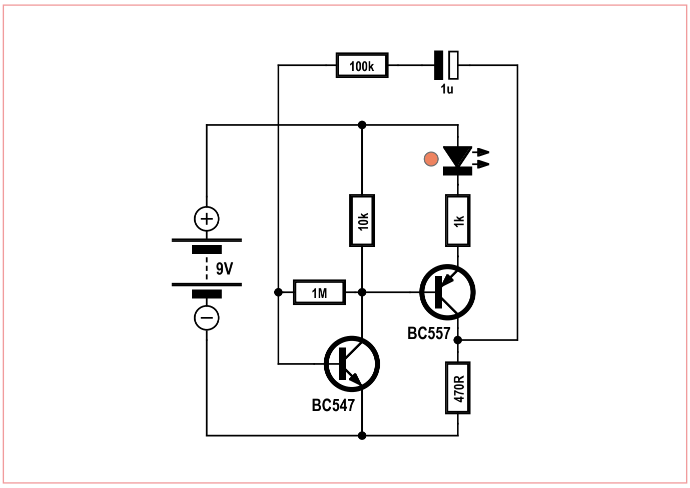


Figure 7. A blinker circuit using complementary transistors.

frequency of the audio signal. Light from a fluorescent lamp produces a raspy tome. Light from a PC monitor also affects the sound, due to modulation of the audio signal at the frame rate.

Voltage to frequency converter

You can use a multivibrator to build a voltage-controlled oscillator (VCO) as illustrated in **Figure 6**. Here both base resistors are connected to a common voltage input. The capacitor charging current, and therefore the oscillator frequency, is directly dependent on the voltage applied to this input. An adjustable voltage divider can be used to set the input voltage to any desired value in order to set the frequency. This circuit can also be used as an acoustic voltmeter with a measuring range of 1 V to 9 V. Among other things, this is handy for quickly checking various batteries.

NPN/PNP flip-flop circuit

You don't necessarily have to use a pair of NPN transistors. **Figure 7** shows a blinker with complementary transistors. As in the previously described circuit, negative feedback from the collector to the base of the NPN transistor establishes a stable operating point. The PNP transistor acts as an emitter follower. A signal with the right phase for the feedback can be taken from the resistor in series with its collector.
The impedance of the feedback path of this circuit is very high. As a result, it has a period of around 1 second with a feedback capacitance of just 1 µF.

Energy-saving LED flasher

In shops you sometimes see advertising signs with a blinking LED that seems to work forever from a single battery. The circuit

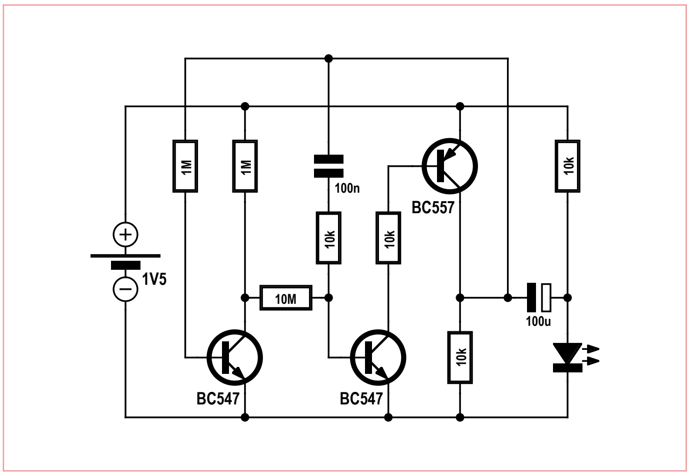


Figure 8. Low-power LED flasher.

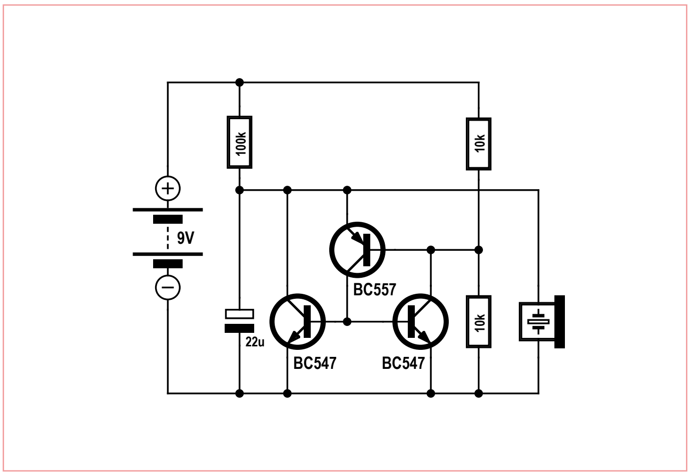
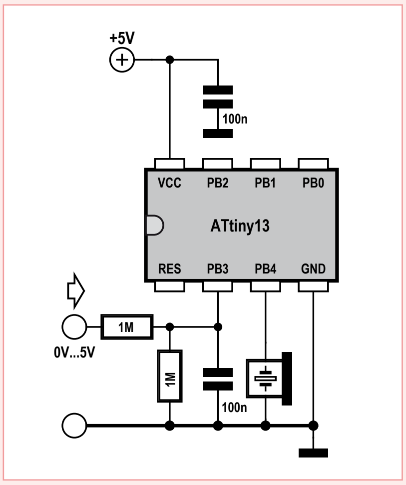


Figure 9. Sawtooth generator.

A voltage to frequency converter with the Tiny13



The voltage to frequency converter described here is really handy because it can be used as an acoustic voltmeter. For instance, a concert A could mean that the battery voltage is OK. Your ears are also good at detecting slow changes. With the microcontroller you can implement a converter with a linear relationship between frequency and input voltage.

Here the ATtiny13 microcontroller operates with a 5 V supply voltage, which means that the measuring range of the A/D converter extends to 5 V. The range is enlarged to 10 V by a high-impedance voltage divider. A piezoelectric acoustic transducer is connected to port B4. The software implements a simple direct digital synthesis (DDS) function with square-wave output. An accumulator A is iteratively incremented by the input sample value until the most significant bit flips, at which point the state of the output signal is toggled. Here the accumulator has a width of 12 bits. At the highest input voltage the output state changes every four measurement intervals, which yields a frequency of just under 600 Hz. The source code can be downloaded from www.elektor.com/120007.

```
'U/f Converter  0...5 V 0...600 Hz
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim U As Word
Dim A As Word

Config Adc = Single , Prescaler = Auto
Start Adc
Ddrb.4 = 1

Do
  U = Getadc(3)
  A = A + U
  A = A And &H0FFF
  If A >= &H0800 Then
    Portb.4 = 1
  Else
    Portb.4 = 0
  End If
Loop

End
```

shown in **Figure 8** is an astable multivibrator with special properties. The 100 µF electrolytic capacitor is charged relatively slowly by a small current and discharged quickly by a short current pulse through the LED. This also generates the necessary voltage boost, since 1.5 V isn't enough for the LED.

This circuit is optimised for low-power operation, which is why the multivibrator circuit is built with a pair of complementary transistors (NPN and PNP). This avoids power losses from control currents. Both transistors conduct only briefly when the LED flashes. To ensure stable operating conditions and reliable oscillation, there is an additional stage with direct-coupled negative feedback. Here again the circuit is designed to work with very high resistance values to minimise power consumption.

The PNP transistor only conducts during the very short pulses occurring every couple of seconds. The output capacitor is charged to nearly 1.5 V between pulses. When the transistor is switched on, the voltage across the capacitor adds to the battery voltage. This produces an open-circuit voltage of nearly 3 V. A red or green LED with a forward voltage of 1.8 V to 2 V connected to the output will flash brightly.

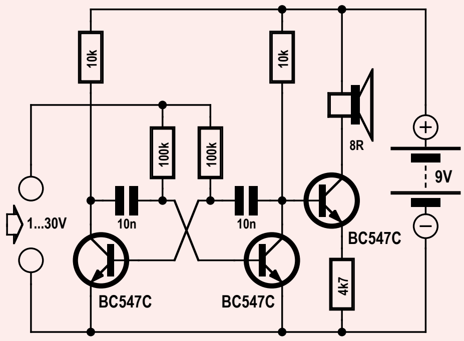
You can use the charging current of the electrolytic capacitor to estimate the current consumption. The average voltage across the pair of charging resistors, each with a value of 10 kΩ, is 1 V. This means that the average charging current is 50 µA. Exactly the same amount of charge is additionally drawn from the battery during the LED pulse. so the average current is around 100 µA. If you assume a battery capacity of 2,000 mAh, the battery should last approximately 20,000 hours, which is over 2 years.

Sawtooth generator

Sawtooth signals, with their characteristic jagged waveforms, can be generated by periodically charging a capacitor to a specific voltage and then discharging it suddenly. This is illustrated in **Figure 9**. While the capacitor is being charged, the PNP transistor is cut off and no base current flows to the two NPN transistors. The discharge level is set to around 5.1 V (4.5 V + 0.6 V) by a voltage divider consisting of two 10 kΩ resistors; above this level the voltage on the base is 0.6 V lower than the voltage on the emitter. This means that the transistor starts to conduct when the voltage rises above 5.1 V, and this current is amplified to obtain a hefty discharge current. This causes the voltage

Quiz

You want to use a simple voltage-controlled multivibrator as an acoustic voltmeter with an 8-ohm loudspeaker. With a supply voltage of 9 V and a corresponding measuring range of 9 V, you use an oscilloscope to check out the circuit. At the moment when one of the transistors of the multivibrator starts to conduct, the base voltage of the other transistor drops to -9 V. After this the 10 nF capacitor is charged by the current through the 100 kΩ resistor to around +0.6 V in approximately 0.65 ms, at which point the circuit switches states.



1) What is the output frequency with a measured voltage of +9 V?

A Approximately 3.3 kHz

B Approximately 330 Hz

C Approximately 770 Hz

2 How does the frequency change when the supply voltage drops but the measured voltage remains the same?

D The frequency drops.

E The frequency remains the same.

F The frequency rises.

3 If you want to increase the volume, what can you change?

G Reduce the value of the 4.7 kΩ emitter resistor of the right-hand transistor.

H Increase the resistor value.

I Replace the emitter resistor by a 100 µF electrolytic capacitor.

If you send us the correct answers, you have a chance of winning a

Minty Geek Electronics 101 Kit.

Send you answer code (composed of a series of three letters corresponding to your selected answers) by e-mail to **basics@elektor.com**.

Please enter only the answer code in the Subject line.

The deadline for submitting answers is 30 September 2012.

All decisions are final. Employees of the publishing companies forming part of the Elektor International Media group of companies and their family member are not eligible to participate.

The correct solution code for the quiz in the May 2012 issue is 'ADH'. Here are the explanations:

Answer 1:

The BF245 provides a constant current of approximately 10 mA, so choice A is correct.

Answer 2:

With a higher input voltage the current will rise quickly with a series resistor, leading to high heat dissipation. By contrast, the FET holds the current constant, so the dissipation is less. Choice D is correct.

Answer 3:

The electrolytic capacitor buffers the voltage under conditions of rapidly changing load current and improves regulation at high frequencies. For example, at 1 kHz the capacitor has an impedance of 1.6 Ω, which reduces the internal impedance of the voltage stabiliser. It makes a noticeable contribution to maintaining the voltage for 1 ms, but not much longer. Choice H is correct.

to drop to 0.6 V, at which point the transistors are cut off and the next charging cycle begins. The circuit shown in Figure 9 has three transistors and is designed for very slow charging. It produces a sort of metronome signal: tick, tick, tick, ...

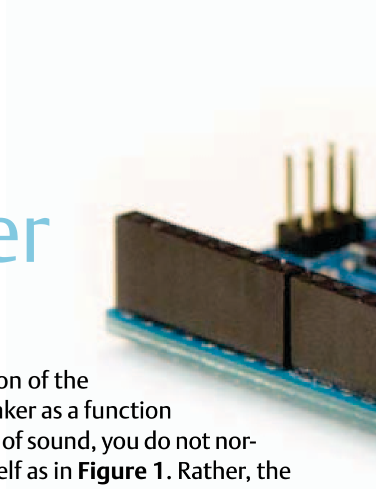
This circuit can be simplified somewhat by omitting the left-hand transistor. In many cases the resulting circuit still works properly, but sometimes there are difficulties with switching off this 'DIY SCR' (see the previous instalment for more information). If the charging current is low, the circuit may get stuck in the on state. This doesn't happen with the three-transistor version, which works reliably over a wide range of charging currents.

Another simplification is also possible. As the piezoelectric transducer is effectively a capacitor, you can omit the electrolytic capacitor. This converts the otherwise slow clock generator into a fast audio generator.

(120007-I)

Arduino on Course (1b)

Part 1b: an Arduino sound player



By David Cuartielles (Spain)

1-bit Sound Generation ... What?!?

Up to this point, you have been experimenting with code that can be generated directly out of the Arduino language. Now we proceed with methods of hacking the processor on the Arduino Uno (the ATmega 328) at a low level, aiming to create a block of code that will use sound data coming from a WAV file for storage inside program memory — and playing back of course.
The method used is not trivial. A good way to start is to look at sound based in terms of its spectrum rather than from any sort of representation in time.

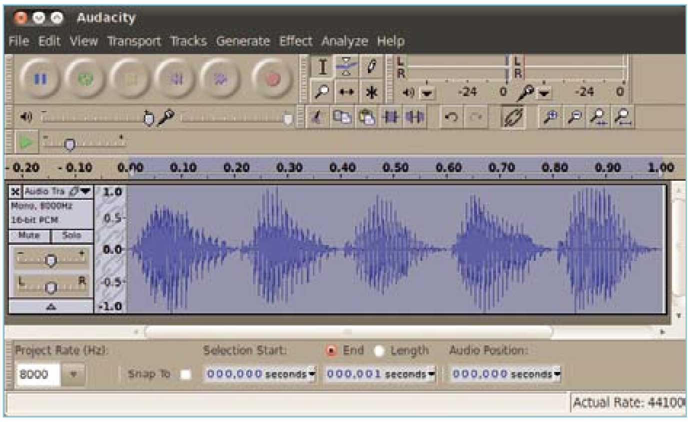


Figure 1. Recording of human speech saying "ta-te-ti-to-tu".

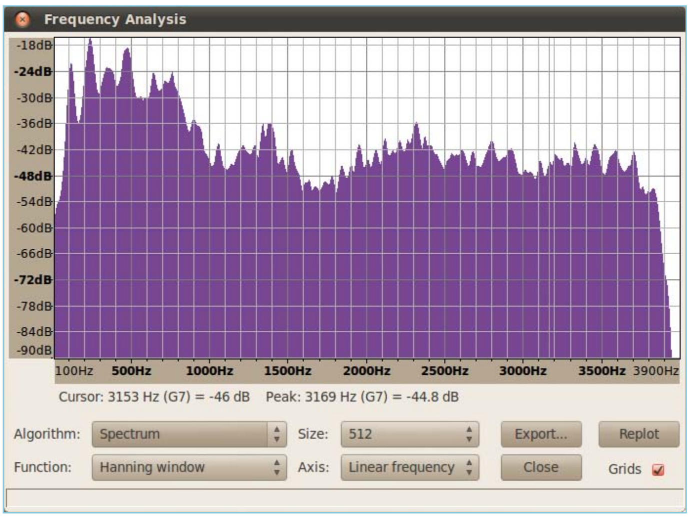


Figure 2. Spectrum of human speech saying "ta-te-ti-to-tu".

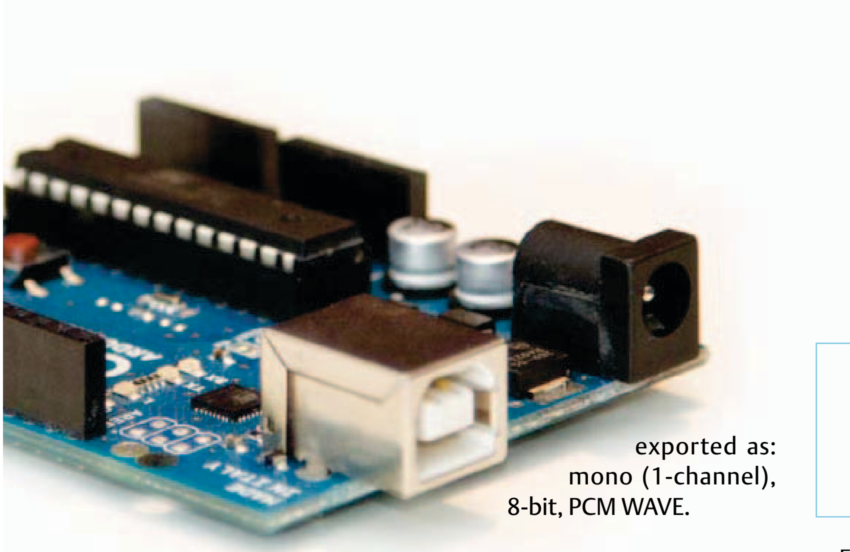
Sound spectrum

The sound spectrum is a representation of the energy transmitted by, say, a loudspeaker as a function of frequency. Looking at the spectrum of sound, you do not normally get a clear view of the sound itself as in **Figure 1**. Rather, the graph represents the amount of energy for different discrete frequencies within a certain amount of time. Typically we represent the spectrum for a whole song (see **Figure 2**).
Alternatively, we can look at the spectrum of just 0.5 seconds of a song. The smaller the time frame we're watching, the better the spectral representation of the sound generated at that very instant. The shape of the wave generating that spectrum can be anything. As a matter of fact, two different sound signals can generate very similar spectra.
Thus, the size of the time frame determines the similarity of that signal is to the original one. The human ear is stimulated by the energy content of the sound, therefore two signals having identical spectra will be perceived as the same, but only if the time resolution is small enough. This is key to the whole science of sound compression: the ability of getting signals that are 'good enough' for us to understand the sound, even if the sound is very different from the original one.
This is also the way 1-bit sound generation works [1]. We can generate sound by having a pulse width modulated (PWM) signal whose average energy level is similar enough to the one of the original signal.
This mathematical trick allows generating medium-quality sound by a microcontroller. The following paragraphs show how to take advantage of this. We will start from a WAV file that can be recorded with your computer. Next, we'll filter it, and transform it into an array of numbers for storing inside Arduino.

Optimal digitalisation and filtering

Many different tools exist for recording sounds. I can only recommend the use of Audacity [2], an open source and free software tool that provides most of the options needed to reproduce sound with microcontrollers (Figures 1 and 2 are screenshots from Audacity).
Before you move on, you should filter the sound. I use a low-pass filter with a 4 kHz roll-off frequency. The Arduino sound player shown here uses a sampling frequency of 8 kHz (i.e. 8000 samples per second), which means that if there were sound components above 4 kHz in your original file, you would hear artefacts in the sound.
With microcontrollers it is possible to reproduce reasonable quality sound, however these chips are limited in terms of memory space. Consequently you need to use sound formats of lower quality that will allow generating several seconds of sound without the use of external memory chips.

A sound format that can be of great use is the 8-bit PCM WAVE (also called Microsoft unsigned 8 bit). This sound format is of sufficient quality to reproduce, for instance, recorded human voice. Thus, the file you'll need in the following step of the process should be



exported as: mono (1-channel), 8-bit, PCM WAVE.

Converting sound to text

Let's look at importing your sound file as a header file that you can add to your Arduino sketch. Since the ATmega 328 microcontroller has 32 KB of Flash memory space, you can use part of that space to store the sound file. You can store large chunks of data into memory arrays by using the `Progmem` library from Atmel's toolchain. 8-bit

```
const unsigned char sounddata_data[] PROGMEM = {128,
    128, 128,
[...]
69, 62, 59, 57, 52, 50, 56, 65, 74, 86, 96, 109, 116, };
```

sound is nothing but a stream of numbers between 0 and 255. It can be declared like:

I have created a tool for Arduino's IDE that enables you to open WAV files and import them directly as part of your code. You can download it from the link mentioned in the reference list, and add it to your sketchbook folder. Place the file into your Arduino sketchbook folder and uncompress it there.

It will add the following folder to your sketchbook: `tools/SoundData`

After rebooting the IDE, you will see a new item under the Tools menu called SoundData. Clicking on it produces a dialog window enabling you to select a WAV file (see **Figure 3**). The second button, titled `Generate Code` will open the WAV file, check whether it is properly encoded, and add a new tab to your code titled

```
// soundData for your Arduino sound project
// automatically generated data
// - original sound file:/development/tmp/matis.wav
// - sampleRate:8000
// - encoding:8 bits
// - channels:1

const int sounddata_length = 7969;

const signed char sounddata_data[] PROGMEM = {127, 127,
    127, 127, 127, 127,
[...]
69, 62, 59, 57, 52, 50, 56, 65, 74, 86, 96, 109, 116, };
```

`sounddata.h`. This new file contains everything you need to play your WAV file as a 1-bit sound. The file will look like this:

But keep on reading before pressing the `Generate Code` button on the dialog window, because there is more to it!

The Sound Player

Figure 3. Dialog Window to import WAV files into your Arduino sketches as header files.

Playing back a sampled sound is not obvious, since it requires changing some of the features of the Arduino core works. There exist libraries for Arduino Uno that hide all of the complexity of making this type of sound player. However, I want you to take a chance and see how this is done at low level, like in **Figure 4**.

The trick that can get Arduino to play a sound sample is the so-called Fast PWM, a feature of Atmel's Atmega family (other brands have it as well, but Arduino Uno runs on Atmel chips). There is a register that allows running PWM at the amazing rate of up to half the clock speed. This allows nice things to be done like playing sound with 1-bit outputs. The only limitation of Fast PWM is that it operates at a resolution of 8 bits only. That's why you should encode your sound files at 8 bits.

To get it all to work, you use two out of the three internal timer registries inside the processor:

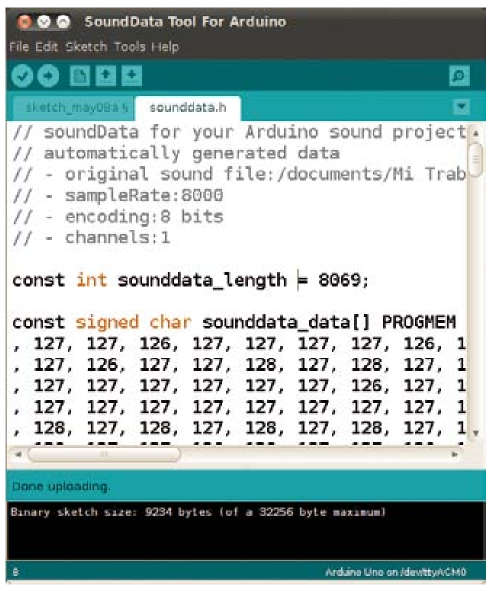


Figure 4. View of the "ta-te-ti-to-tu" sound file within the Arduino IDE after importing it with the SoundData tool.

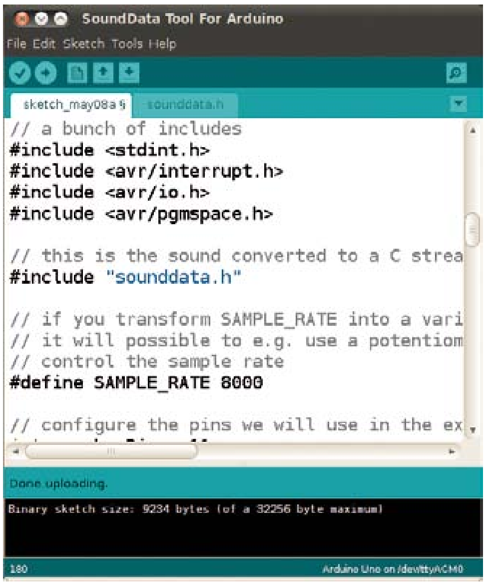


Figure 5. Screendump of the basic sound player after automatically generating the code.

- The first clock operates at SAMPLE_RATE and is used to read the next value from the computer generated `sounddata_data` array. This value sets up the duty cycle for the PWM stream being output through pin 11.
- The second timer ticks at 62500 times per second (16000000 / 256), i.e. much faster than the playback rate (default SAMPLE_RATE is 8 kHz). As explained earlier, this generates a digital signal whose spectrum is very similar in shape to that of the original sound file. In other words, by means of increasing the PWM frequency, the generated signal gets a spectrum that resembles the one of the sound. In many applications this is more than enough to play sound at a low price.

Note: by adding this code to your Arduino sketch you will be overriding some of the basic commands of the Arduino language. E.g. the PWM will stop working at some of the pins, and the `delay()` function will not operate as expected. This will only affect this sketch.
The first time this technique was shown in the Arduino world was back in 2008. Michael Smith published an example on the Arduino playground that would play the sound of a MAC computer booting up, upon reset of the Arduino Diecimila board. Michael's code was based upon the work of many others which I have included as references [3].

Just to make it easy, the SoundData tool will not just generate the sound information, but also include the code to the basic sound-playback example inside your sketch. Beware, make sure your sketch is empty when calling the tool, as it will overwrite anything on your IDE.
The basic program to play 1-bit sound (**Figure 5**) includes a lot of

```
ISR(TIMER1_COMPA_vect) {
    if (sample >= sounddata_length) {
        if (sample == sounddata_length + lastSample) {
            // this is the condition of reaching the last
sample
            stopPlayback();
        } else {
            // Ramp down to zero to reduce the click at
the end of playback.
            OCR2A = sounddata_length + lastSample
- sample;
        }
    }
    else {
        // OCR2A is the register in memory that will
push
        // PWM at high frequency to pin 11
        // pgm_read_byte reads data out arrays stored in
program memory
        OCR2A = pgm_read_byte(&sounddata_data[sample]);
    }

    // increase the sample count
    ++sample;
}
```

low level commands in order to override the timers. I will describe the functions one by one. Let's start with `TIMER1`:
`ISR` is the name for the interrupt handling function inside the microcontroller. An interrupt is an event that will tell the chip to stop doing anything it is into at the time and attend a certain event. Processors can have both internal and external interrupts. The internal ones are timers, while the external ones happen when certain pins change from `HIGH` to `LOW` or vice versa.
This one instance of the `ISR` function is taking care of the arrival of internal `TIMER1` events. Every time `TIMER1` ticks, this function will do the following:

- increase the counter used to address the sound data;
- check whether the end of the sound data array is reached;
- if not at the end, load the next sample from the array;
- if at the end, fade the sound to zero.

```
void startPlayback()
{
    pinMode(speakerPin, OUTPUT);

    // Set up Timer 2 to do pulse width modulation on
the speaker
    // pin.

    // Use internal clock (datasheet p.160)
    ASSR &= ~(_BV(EXCLK) | _BV(AS2));
```

```
    // Set fast PWM mode  (p.157)
    TCCR2A |= _BV(WGM21) | _BV(WGM20);
    TCCR2B &= ~_BV(WGM22);

    // Do non-inverting PWM on pin OC2A (p.155)
    // On the Arduino this is pin 11.
    TCCR2A = (TCCR2A | _BV(COM2A1)) & ~_BV(COM2A0);
    TCCR2A &= ~(_BV(COM2B1) | _BV(COM2B0));

    // No prescaler (p.158)
    TCCR2B = (TCCR2B & ~(_BV(CS12) | _BV(CS11))) |
   _BV(CS10);

    // Set initial pulse width to the first sample.
    OCR2A = pgm_read_byte(&sounddata_data[0]);

    // Set up Timer 1 to send a sample every interrupt.
    cli();

    // Set CTC mode (Clear Timer on Compare Match)
   (p.133)
    // Have to set OCR1A *after*, otherwise it gets
   reset to 0!
    TCCR1B = (TCCR1B & ~_BV(WGM13)) | _BV(WGM12);
    TCCR1A = TCCR1A & ~(_BV(WGM11) | _BV(WGM10));

    // No prescaler (p.134)
    TCCR1B = (TCCR1B & ~(_BV(CS12) | _BV(CS11))) |
   _BV(CS10);

    // Set the compare register (OCR1A).
    // OCR1A is a 16-bit register, so we have to do this
   with
    // interrupts disabled to be safe.
    OCR1A = F_CPU / SAMPLE_RATE;    // 16e6 / 8000 =
   2000

    // Enable interrupt when TCNT1 == OCR1A (p.136)
    TIMSK1 |= _BV(OCIE1A);

    lastSample =
   pgm_read_byte(&sounddata_data[sounddata_length-1]);
    sample = 0;
    sei();
}
```

Both timers TIMER1 and TIMER2 are initialized as part of the startPlayback function. Let’s see how what looks like:

Although the sequence of low level commands is explained in the code, a summary of it may be useful for the non-expert:
- turn the pin for the speaker into an output;
- configure the board to use the internal clock for this;
- initialise Fast PWM mode;

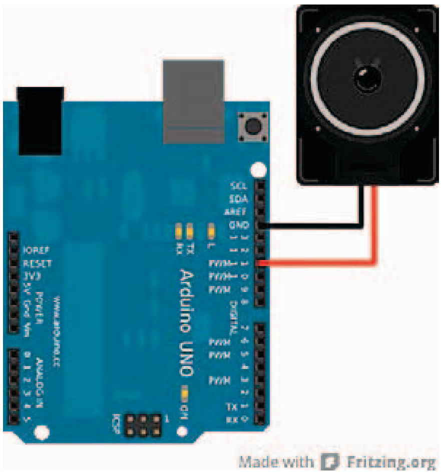


Figure 6. Arduino Uno is perfectly capable of driving a small loudspeaker directly.

- configure `TIMER2` to run the PWM that will play the sound, the way to set the duty cycle to the PWM is by changing the value of the register called `OCR2A`;
- load the first sound sample into `OCR2A`;
- stop the interrupts for a second – `cli()` – so that we can configure the `TIMER1` without breaks;
- configure `TIMER1` to tick for picking up the next sample;
- load the last sound sample;
- restart the interrupts – `sei()`.

```
void stopPlayback()
{
    // Disable playback per-sample interrupt.
    TIMSK1 &= ~_BV(OCIE1A);

    // Disable the per-sample timer completely.
    TCCR1B &= ~_BV(CS10);

    // Disable the PWM timer.
    TCCR2B &= ~_BV(CS10);

    digitalWrite(speakerPin, LOW);
}
```

In a similar fashion, you need to have a function that will stop the timers of counting this way once the end of the sound is reached. This leaves you with a series of functions you can call anywhere

```
void setup()
{
    startPlayback();
}

void loop()
{
    // do nothing
}
```

from a program to play sound this way. In this case, the example by default will call `startPlayback()` within setup. In this way the sound will be played just once.

Closing words

This article is an introduction to different ways of producing sound using Arduino. You have seen how to play tones by means of basic Arduino functions. Libraries have been described that simplify the way basic melodies can be played using inexpensive piezo buzzers. Finally you got a sneak peek at the programming behind Fast PWM to generate 1-bit sound.

All code chunks and tools discussed here are packed into a ZIP file [4], including properly formatted sound files for you to try all the examples. I also created a new tool for the Arduino IDE that will help you with the importing of short WAVE files into Arduino's program memory. The tool enables you to load sounds and play them back, change their sample rate, play them backwards or scratch the sound. In terms of hardware, you connect your loudspeaker as shown in **Figure 6**.

But don't stop here! There is a lot to explore — for instance using pin 3 in parallel with pin 11 to produce stereo sound. Or create 8-bit synthesizers with the ability of mixing four sound lines on a single channel. What about transforming your Arduino board into a MIDI activated soundcard?

Happy hacking. See you in a month.

(120427)

References

[1] **1-bit Sigma Delta DA Converters:**
www.digitalsignallabs.com/presentation.pdf

[2] **Audacity, the free and open source sound studio:**
audacity.sourceforge.net

[3] **Various references from the example by Michael Smith:**
Arduino reference on the use of the tone library:
arduino.cc/en/Reference/Tone
Original article on the Arduino Playground:
arduino.cc/playground/Code/PCMAudio
www.uchobby.com/index.php/2007/11/11/arduino-sound-part-1/
www.atmel.com/dyn/resources/prod_documents/doc2542.pdf
www.evilmadscientist.com/article.php/avrdac
http://gonium.net/md/2006/12/27/i-will-think-before-i-code/
http://fly.cc.fer.hr/GDM/articles/sndmus/speaker2.html
www.gamedev.net/reference/articles/article442.asp

[4] **Example files for this article:**
www.elektor.com/120427

AS3935 Lightning Sensor

By Raymond Vermeulen (Elektor Labs)

austriamicrosystems' new lightning sensor IC measuring just 4x4 mm landed on my desk like a stroke of lightning, thanks to a suggestion from my colleague Luc Lemmens. If a suitable antenna is connected to the IC, a microcontroller linked to the device via SPI or I²C can calculate the distance to the edge of a thunderstorm. That's right – not the distance to the lightning, but instead the distance to the storm front. The range is 5 to 40 km (approx. 3 to 25 miles).

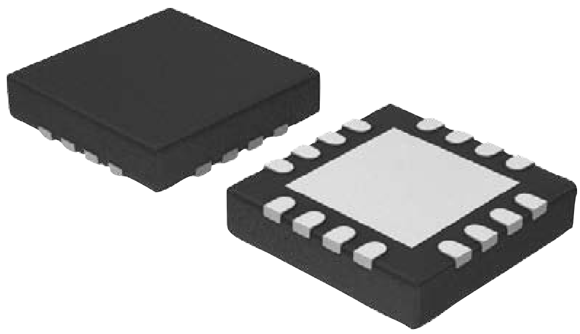
(120405)

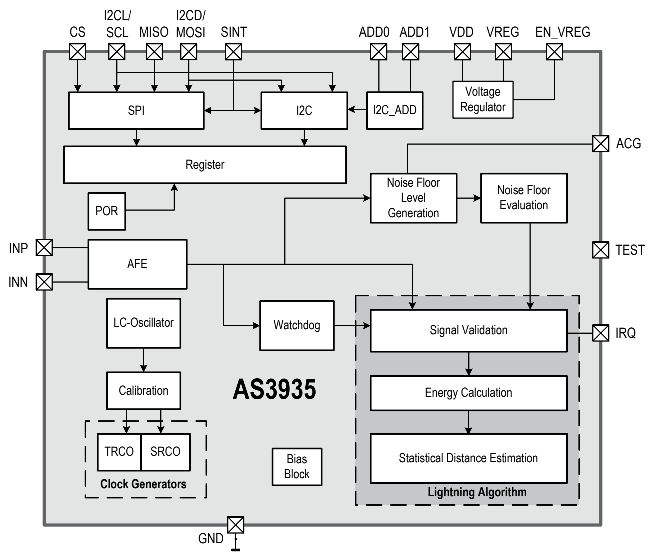
AS3935

The microcontroller connected to the AS3935 can also configure a lot of settings in the IC. One of the really remarkable features of the algorithm used in this IC is that it can determine whether the received signals are man-made or naturally occurring. This means that you can also use it indoors. What's more, the noise floor level can be adjusted in a register to allow the device to be used in a noisy environment. An interrupt is sent to the microcontroller when the received noise level rises above the configured noise floor, which allows the microcontroller to raise the noise floor level if necessary. The gain of the integrated amplifier also has a separate mode for indoor use. Artificially generated noise can adversely affect the lightning detector, so the detection level can also be adjusted in a register, although this reduces the detection sensitivity.

The recommended antenna is a loop antenna consisting of a parallel-resonant LC circuit with a *Q* factor of approximately 15 and a resonant frequency of 500 kHz. As it is rather difficult to make an antenna that perfectly meets these specifications, the IC has provisions for tuning the antenna. There are register settings that allow the microcontroller to read out the resonant frequency of the connected antenna using a signal on the IRQ pin with a configurable division ratio. The microcontroller can tune the antenna circuit by adding 0 to 120 pF of capacitance in steps of 8 pF. Once the antenna is properly tuned, the rest of the internal oscillators in the IC can be calibrated. All interrupts are signalled by setting the IRQ pin high, and the microcontroller can read a register to determine the specific cause of each interrupt. It is also possible to configure the IC to sense a minimum number of discharges in a 15-minute interval if you are not interested in detecting individual sporadic discharges.

As our readers are generally interested in weather-related projects, it wouldn't surprise me to see a circuit based on this device appear in a future issue of Elektor.





Description	Condition	Value
Supply voltage range	EN_VREG = VDD	2,4 – 5,5 V
Supply voltage range	EN_VREG = GND	2,4 – 3,6 V
Current in power-down mode	VREG = OFF	1 µA
Current in listening mode	VREG = OFF	8 µA
Current in signal verification mode		350 µA

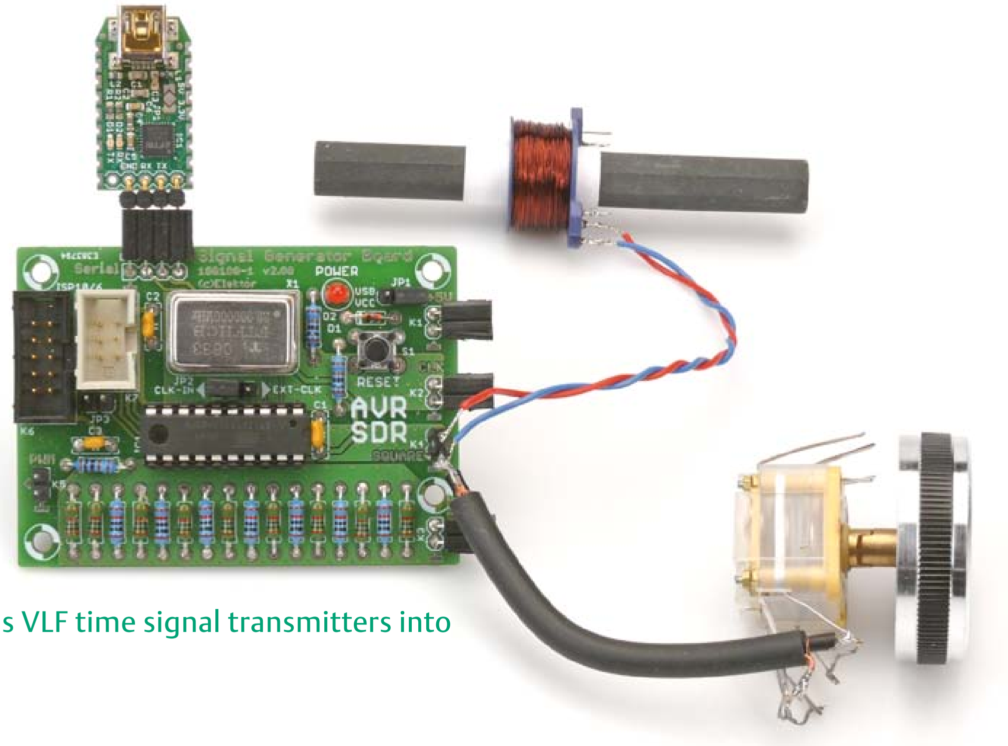
Austriamicrosystems AS3935 data sheet: www1.futureelectronics.com/doc/AUSTRIAMICROSYSTEMS/AS3935.pdf

Evaluation kit application note:
http://media.digikey.com/pdf/Data%20Sheets/Austriamicrosystems%20PDFs/AS3935_EvalManual_AN.pdf

AVR Software Defined Radio (5)

Part 5: Decoding DCF77, MSF and TDF162 using IIR and matched filters

By Martin Ossmann (Germany)



The aim of this series of articles is to show that the popular AVR microcontrollers are also suitable for digital signal processing. In this instalment we use a variety of decoding methods and filters to convert the signals from various VLF time signal transmitters into digital data.

In the previous instalment [4] we devoted our attention to the reception of digital radio signals. Using the digital receiver described in that instalment, we received signals from time signal transmitters such as German DCF77 and from weather services and decoded the received signals. Now it's time to examine a variety of decoding methods. Along with transmitting and receiving RTTY signals, we discuss the reception and decoding of signals from DCF77 in Germany, MSF in the UK and TDF162 in France. Finally, we describe how to use a matched filter for bit decoding. Sadly the author can't even dream of ever receiving WWVB at his location, but the information supplied in this article should enable Elektor USA readers to develop suitable decoding and demodulation algorithms. WWVB transmits at 60 kHz.

Wireless data transmission at 125 kHz

If the level of the DDH signal at your receiver location (see the previous instalment in this series) is too weak or absent altogether, you can build your own test transmitter. The hardware is exactly the same as for the DCF test transmitter described in the third instalment [3]. A simple ferrite antenna and a variable capacitor form a series-resonant circuit that is fed from the square-wave output of the signal generator. This arrangement is shown in the photo at start of the article.

The software for the transmitter is contained in the *EXP-SQTX-FM-RTTY-V01.c* file. After configuring the reception software in *EXP-125kHz-RTTY-RX-V01.c* for operation at 125 kHz, you can start your signal transmission and reception experiments. The author obtained a range of over 5 m (15 ft) (even through walls) with this simple system. Among other things, this arrangement could form the basis for a wireless remote control system.

With the same software, you can also decode conventional RTTY ('telex') signals in the amateur radio bands. All that is necessary for this is to adjust the filters and the timing to match the transmission parameters. For testing, you can generate RTTY signals with suitable PC software such as MMtty (download from [6]) and a sound card.

Decoding DCF77

An oscillogram of the received and demodulated DCF77 signal was shown in Figure 3 of the third instalment of this series. Now let's see how you can recover the data from the keying pulses. As the amplitude of the received signal can vary widely depending on reception conditions, the first thing you need to do is to define a threshold level (as automatically as possible) for deciding whether the instantaneous amplitude is high or low. For the sake of simplicity, you could set the threshold level at half the average signal level. However, this requires determining the long-term average value. If we wanted to use a CIC filter to average the signal, we would need a very large interim buffer. This is why we use an IIR filter [7] instead.

IIR filtering

The abbreviation IIR stands for 'infinite impulse response'. This long impulse

response (which corresponds to the averaging interval) is obtained by using a recursive filter. The computation formula for a simple IIR filter is:

$y_{k+1} = a\,x_k + (1 - a)\,y_k$

Here y_k is the output sample series, x_k is the input sample series, and a is a number less than 1 but close to 1. The new sample y_{k+1} is the averaged sum of the previous sample and the new input sample.
In this formula the factors a and $(1 - a)$ are normally numbers less than 1. This makes it tempting to use floating-point operations to compute the filter, but that would be much to complicated. In our implementation (see **Listing 1**) the filter coefficients are represented as fractions, with the denominator (65,536) being a power of 2.
Here $a = (65536 - 50) / 65536 = 0.999237$ ..., and $(1 - a)$ is accordingly 0.000763... The computations are performed using 32-bit integer numbers. The compiler coverts division by 65,526 into simply selecting the most significant 16 bits, which considerably speeds up the computation. The result is a highly efficient IIR filter. The output of the IIR filter is the signal in binary form, and the time of day can be obtained by evaluating the pulse lengths.
In central Europe and the south-east of the UK, DCF77 reception with decoding requires the simple front-end or the universal receiver, along with the active ferrite antenna tuned to 77.5 kHz. With the *EXP-DCFdecode-RX-V01.c* software loaded in the receiver microcontroller, you obtain a seconds timer and the signal is output by the PWM DAC. The signal strength and time of day are also shown on the LCD module, and the decoded data is output on the RS232 or USB port.

Decoding MSF

There are several time signal and standard frequency transmitters in Europe. One of them is MSF [8] in Rugby, Great Britain, which broadcasts at 60 kHz (coinciding with WWVB in Fort Collins, Colorado, USA). Its signal is relatively weak at the author's place of residence, so a bit more effort is necessary. The amplitude of the MSF carrier wave is keyed at 1-second intervals (see **Figure 1**). The carrier is keyed to a low level for 0.5 s at the start of each minute. At the start of each subsequent second, the carrier is initially keyed low for 0.1 s. The next 0.5 s is the time slot for the A bit. The A bit is high if the carrier amplitude is low in this time slot. The following 0.1 s is the time slot for the B bit. The B bit is also high if the carrier amplitude is low in this interval. The A and B bits are used to transmit time data, additional synchronisation data and error detection codes.

Our first attempts to receive the signal yielded a severely distorted signal, which could not be adequately filtered by a simple CIC filter. We therefore needed a better filter with a cutoff frequency at around 10 kHz and fairly strong attenuation. Another requirement for the filter was that it should not affect the waveform too much. These characteristics can be achieved with a FIR filter [9] of sufficient order.

For MSF reception you need the receiver, the active antenna tuned to 60 kHz, and the software *EXP-RX-MSF60decode-V01.c*. With this arrangement, the signal strength and time of day are shown on the LCD module and the date and time are output on the serial port. The structure of the fifth-order FIR filter is shown in **Figure 2**. The input

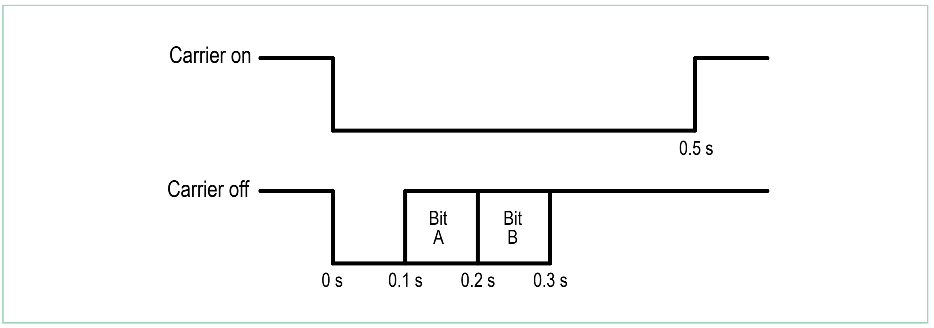


Figure 1. Carrier keying of the Rugby MSF signal.

Listing 1: IIR filter

```
#define ALPHA 50
#define BETA  65536
int32_t Threshold ;

  Threshold=(ALPHA*(int32_t)Amplitude+(BETA-ALPHA)*Threshold)/BETA ;
  if (Amplitude>Threshold) {
    DCFlevel=0 ;
    }
   else {
    DCFlevel=1 ;
    }
```

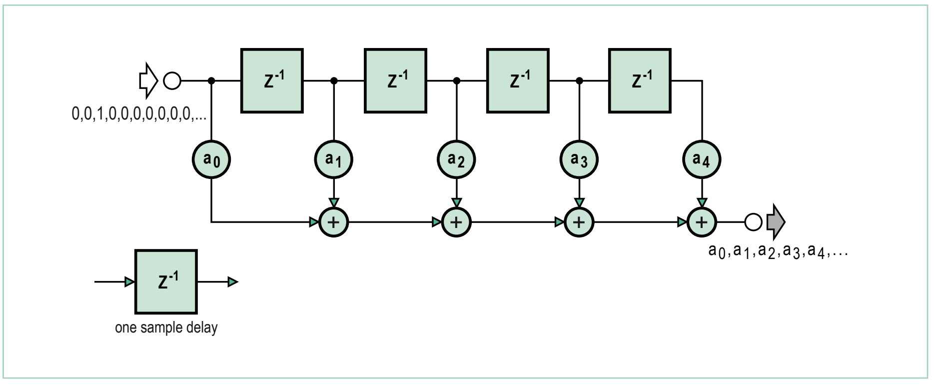


Figure 2. Signal paths in an FIR filter.

Listing 2: FIR summation

```
#define FIRLength 60

prog_int16_t FIRCoeffs []={
   112,      3,    -17,
   -52, ............ 112 }  ;

  FIRSum=0 ;
  j=FIRPointer ;
  for (k=0 ; k<FIRLength ; k++ ) {
    FIRcoef=pgm_read_word( FIRCoeffs1 + k ) ;
    FIRSum += (int32_t)FIRcoef*(int32_t)FIRBuffer[j] ;
    j++ ; if (j==FIRLength) { j=0 ; }
    }
  return FIRSum/0x10000L ;
```

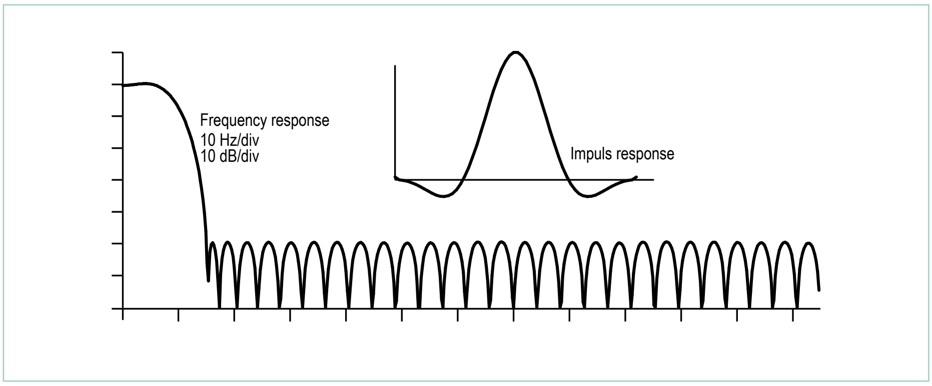


Figure 3. Impulse response and frequency response of the FIR filter.

Figure 4. Demodulated and filtered MSF signal and blue timing trace.

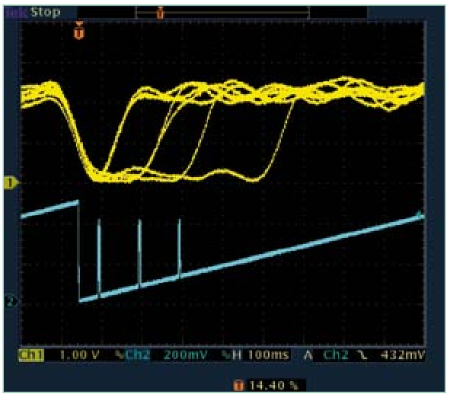


Figure 5. The three spikes show where the signal is evaluated.

samples pass through a buffer, and the output sample is the sum of the samples in the buffer weighted by the coefficients *k*.

All of this is implemented in the C code shown in **Listing 2**. Here again we use fast integer arithmetic followed by division. This routine filters the signal, which is sampled at a rate of 250 samples per second. The computation time (300 µs) is sufficiently short for a filter order up to 60, since 4 ms is available for each sample in this case.

Figure 3 shows the impulse response and frequency response of the filter. The initial cutoff frequency is approximately 8 kHz, and the attenuation above 15 kHz is around 50 dB.

Figure 4 shows the demodulated and filtered MSF signal (yellow trace). The minutes pulse with a duration of 0.5 s can be seen near the middle of the image. A pulse with A = 1 and B = 0 can be seen two seconds before the minutes pulse, and a pulse with A = 0 and B = 1 can be seen after the minutes pulse. This is followed by several pulses with A = 0 and B = 0. A trace with a sawtooth waveform is visible below the yellow trace. This signal can be used to keep track of the timing.

The waveforms are shown in more detail in **Figure 5**. The software has a variable called *SecondTimer*, which recurrently counts from 0 to 250 at a rate of 250 Hz. This clock rate is the same as the sampling rate of the demodulated signal. *SecondTimer* is reset when the minutes pulse is detected, so it always starts counting from the start of a second. The *SecondTimer* count state is output by one of the PWM DACs. The three blue spikes show where the software evaluates the received signal. This is where the values of the A and B bits are determined. After all the bits have been collected, they are evaluated and the time data is output to the LCD module and on the RS232 port. **Listing 3** shows an example of the output data.

Decoding France Inter (TDF)

France Inter (TDF) also broadcasts time data using a phase-modulated carrier. The signal waveforms for the low and high states are shown in **Figure 6**. The data in

Listing 3: Example data from the MSF transmitter

```
<2000000000000000000100000001100100000010001110001001011111103>  07:09 08.03.10 MONDAY    P=1 P=1 P=1 P=1
<2000000000000000000100000001100100000010001110010000011113103>  07:10 08.03.10 MONDAY    P=1 P=1 P=1 P=1
<2000000000000000000100000001100100000010001110010001011111103>  07:11 08.03.10 MONDAY    P=1 P=1 P=1 P=1
```

this signal is coded in the same way as the DCF77 signal. However, we encounter a new problem with the decoding. The time code bit is transmitted at the start of each second, but this is followed by a lot of phase-modulated data that is used for internal purposes. This means that the receiver must first determine the exact position of the seconds pulse. There is a long modulation gap before the seconds pulse for the null second, since the pulse for the 59th second is mission. Once the receiver has detected this gap, it looks for the next peak in the phase waveform (see Figure 6). This peak marks the exact position of the desired seconds pulse. At exactly one-second intervals from this point onward, the receiver checks whether the received signal indicates a 1 or a 0.

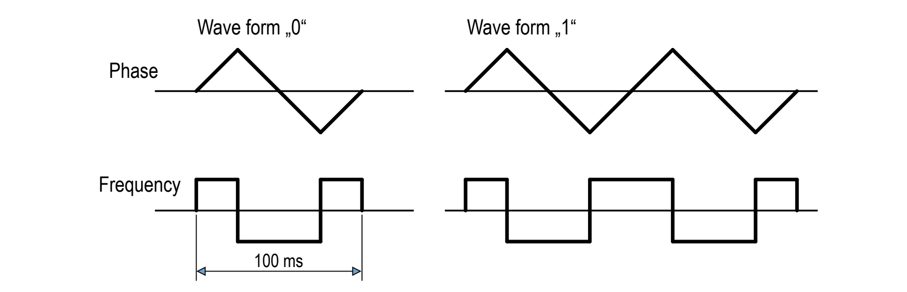


Figure 6. TDF162 signal waveforms.

A matched filter with CIC

If you want to evaluate the phase directly, you don't want to have any phase drift. Consequently, you need to use a PLL to stabilise the phase of the signal. However, this is a bit complicated and control loops are not always stable, so a different approach is better. Instead of evaluating the phase, we evaluate the frequency. The frequency can be obtained from the phase by differentiating successive samples and filtering the resulting samples with a CIC low-pass filter to clean up the waveform. To determine whether a 0 or a 1 has been transmitted, we check whether the waveform corresponding to a 1 has been received at the right time. If it has not, we conclude that a 0 has been received.

This leaves us with the question of how to determine whether the signal waveform we are looking for has actually been received. A matched filter is often used for this purpose in telecom engineering. The impulse response of such a filter exactly matches the desired waveform (although it is mirrored in time). Our 0 waveform consists of a combination of three rectangular pulses. This makes it fairly easy to implement a matched filter in the form of a CIC filter, since the impulse response of such a filter — the response to a single 1 at the input — is naturally a single rectangular pulse, as described in [4].

The filter is shown schematically in **Figure 7**. Here the signal is sampled at a rate of 500 samples per second. The first pulse has a length of 25 ms and therefore corresponds to twelve samples. The second pulse has a length of 50 ms and corresponds to 25 samples. The final pulse, like the first one, has a length of 25 ms. These pulse lengths translate directly into the required delay stages. This filter is ideal for detecting the 0 waveform. As the 1 waveform is similar to the 0 waveform (consisting of two successive 0 waveforms), we can use the same filter and simply check the output 50 ms later. This is exactly the strategy that is used in the receiver.

The oscillograms (**Figures 8** and **9**) show the corresponding waveforms. The yellow trace shows the state of the timer that

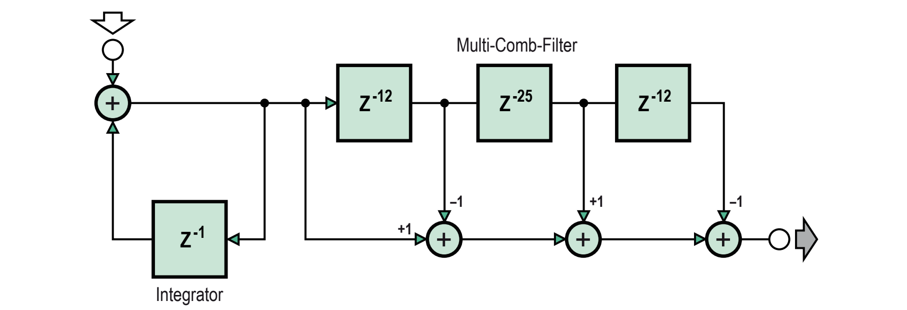


Figure 7. A matched filter implemented as a CIC filter.

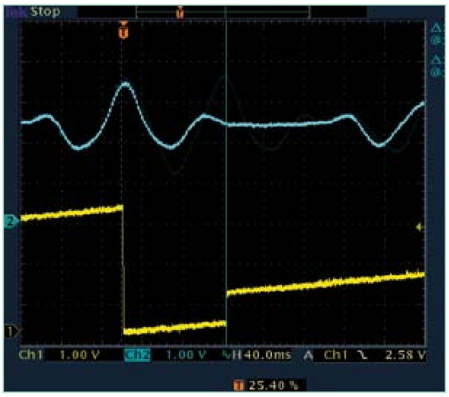


Figure 8. The output signal of the matched filter when a 0 is received.

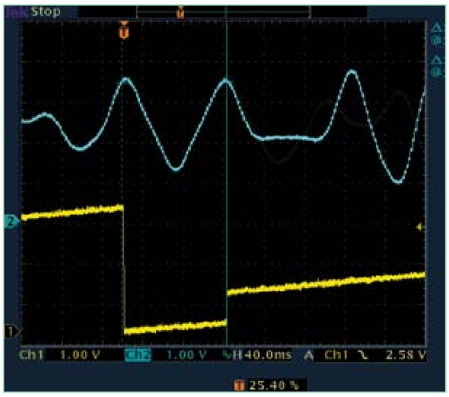


Figure 9. The output signal of the matched filter when a 1 is received.

detects the gap for the 59th second. The input signal is sampled at the points where there is a sudden change in the yellow signal, which is exactly where the peaks occur in the received signal.

The zeros and ones are stored in an array. After the final bit is received, the data in this array is decoded and displayed on the LCD module. It is also output at the same time on the serial port at 19,200 baud. **Listing 4** shows an example of the received data.

The final instalment of this series will appear in the next edition. In that instalment we will look at bit clock synchronisation and describe the 'early late gate synchroniser', among other things.

Finally, we will decode the signal from the BBC198 Droitwich transmitter and get acquainted with several other decoding methods.

(120089-I)

Listing 4: Example data from TDF162

```
TDF 162
0001110000000000001011001100100010011110101100100000000100000: 08:19 17.02 WEDNESDAY 2010 p000
0000110000000000001010000010100010011110101100100000000100000: 08:20 17.02 WEDNESDAY 2010 p000
0000110000000000001011000010000010011110101100100000000100000: 08:21 17.02 WEDNESDAY 2010 p000
0000110000000000001010100010000010011110101100100000000100000: 08:22 17.02 WEDNESDAY 2010 p000
0001110000000000001011100010100010011110101100100000000100000: 08:23 17.02 WEDNESDAY 2010 p000
```

Internet Links

[1] www.elektor.com/100180

[2] www.elektor.com/100181

[3] www.elektor.com/100182

[4] www.elektor.com/120088

[5] www.elektor.com/120089

[6] http://hamsoft.ca/pages/mmtty.php

[7] www.dspguru.com/dsp/faqs/iir/basics

[8] www.npl.co.uk/science-technology/time-frequency/time/products-and-services/msf-radio-time-signal

[9] www.dspguru.com/dsp/faqs/fir/basics

BasicCard goes contactless
A discreet alternative

By Patrick Gueulle (France)

The only chip card that you can program in BASIC has now been on the market for more than ten years. It remains under continuous development: in a new twist, this well-known card with an open operating system is now available in an RFID version. As well as the new facilities for contactless operation, very powerful (and free!) development tools are now available to provide an easy way to get to grips with this fascinating technology.

The philosophy behind the product has not changed since the first 'Compact' BasicCard appeared in 1998: put into the hands of developers (which includes interested enthusiasts) the means to develop their own applications independently of the mass-market chip card manufacturers. The result is a complete range of asynchronous cards using Flash technology, with products available in small quantities and even individually. With the help of a (re-)programmable virtual machine, which in some versions was even capable of supporting several applications simultaneously, there is support for a high-level language simpler, but no less powerful, than Java: ZCBasic (**Figure 1**).

A complete development environment (compiler, simulator and double-debugger, along with a manual running to some 250 pages) is available for free download at [1].

With just a couple of lines of source code it is possible to make a BasicCard compatible with just about any terminal, including for example a mobile phone. This existing know-how can now be transferred to contactless applications using the ZC7.5 RFID version of the card. The transition is also simplified by the use of the ZC7.5 Combi card, which offers two interfaces: one over contacts (transport layer protocol T=0 or T=1) and one contactless (ISO 14443 type A T=CL). An ACR122 or Omnikey 5321 makes a suitable RFID reader for the contactless interface.

Card applications

Twenty or so lines of code (RFIDspy.BAS [2]) suffice to demonstrate the flexibility of the ZC7.5 Combi card. At under 400 bytes of P-code the program occupies just a tiny fraction of the 32 kbyte EEPROM space in the powerful IC. This short program is a T=CL version of the logger that we presented for the first T=0 BasicCard with contacts in the May 2002 issue of *Elektor* [3] (for card version ZC4.1). The program can store and subsequently dump the commands used by a reader as part of its dialogue with a card presented to it. Depending on the reader, this 'impostor' will either be rejected almost instantly or be accepted (for a while at least) by the reader as a genuine card intended for use with it.

The underlying idea, of course, is to use this code as a basis for incremental extension: as more and more of the commands emitted by the reading device are understood, code can be written to deal more precisely with them, thus better emulating a genuine card. There is a certain amount of detective work to be done in unmasking the security mechanisms used in the design and in determining their strengths and weaknesses. In summary, we have a very handy experimental tool.

Skipping over the first three lines of the source code, which are just preparatory declarations, we come to a series of '#Pragma' directives, of which the first two are specific to operation in con-

tactless mode. The first thing to know is that most contactless objects (cards and tags) have a unique number (or UID), several bytes long, which is written into ROM during manufacture. This is occasionally used as part of a defence against cloning, but its main use is during the anti-collision process used by the reader to communicate with a single specific card when more than one is within its range. The details of this process are relatively complex, but with luck card and reader will handle it all themselves: unless you really want to get his hands dirty, as an applications programmer you need not get involved. However, you can specify the number of bytes that will be used to form the UID in order to match the characteristics of another card as closely as possible. The ZC7.5 supports three standard variants: 'single' (four bytes, like MIFARE Classic), 'double' (seven bytes, like MIFARE Ultralight), and 'triple' (ten bytes). It is also possible to replace the fixed UID in the chip with a random value to help the owner of a card to avoid being tracked. In our example the UID is sent out as a group of four random bytes when a reader starts polling:

```
#Pragma UID(Random,Single)
```

At the beginning of communication begins between reader and card, the reader selects it and waits for its reply (ATS for 'answer to select'), which is comparable to ATR ('answer to reset') in the case of a card with contacts.

```
#Pragma ATS(TA1=0,FWI=7,TC1=0,HB="EMVA")
```

This second command allows the default communications parameters to be modified, either partially or completely, in order to optimise compatibility with a particular reader. In this example we choose compatibility with the 'EMV Contactless Specifications' (which are publicly available: [4]). These specifications ensure compatibility for electronic payments between chip cards and terminals that bear a special logo indicating that they comply with them (**Figure 2**).

In a similar way we can modify the 'ATR' parameters of the card, which affect its communications over the contact interface when it is connected to a suitable reader:

```
#Pragma ATR(Direct,T=1,HB="RFIDspy")
```

In this case we specify use of the T=1 protocol and select 'direct convention' for communications; we could equally well have used the T=0 protocol and/or 'inverse convention'. We will now look at what happens in the card when it is selected by the reader. The BasicCard has an internal file system, similar to MS-DOS. Opening a file called, for example, 'Card.Log', can be done as follows:

```
Open "Card.Log" For Append As #1
```

To allow this file to be deleted using an external instruction, it is convenient to define a special command for the purpose. Here we

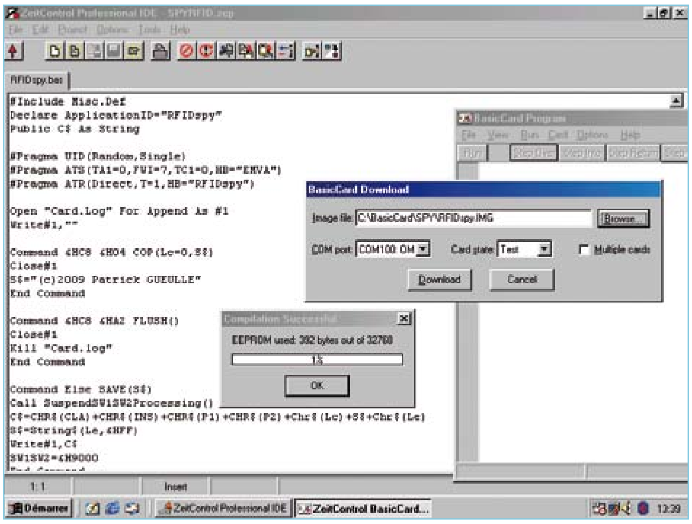


Figure 1. The ZeitControl BasicCard development environment in action.

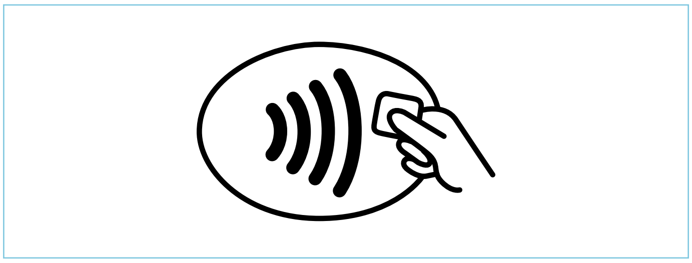


Figure 2. The *EMV contactless* logo is found on an increasing number of point-of-sale terminals.

have called the command 'FLUSH':

```
Command &HC8 &HA2 FLUSH()
Close
Kill "Card.Log"
End Command
```

Now, if we send the byte sequence `C8 A2 00 00 00` to the card it will delete the file. The main part of the program is found in the next seven lines:

```
Command Else SAVE(S$)
Call SuspendSW1SW2Processing()
C$=CHR$(CLA)+CHR$(INS)+CHR$(P1)+CHR$(P2)+Chr$(Lc)+Chr$(L
e)+S$
S$=String$(Le,&HFF)
Write#1,C$
SW1SW2=&H9000
End Command
```

That is all that is needed to trap any unrecognised command (hence the 'Else') received by the card and store it in the file Card.Log along with the parameters CLA, INS, P1, P2, Lc, Le and any data received from the terminal ('incoming' commands).

For the outgoing message the card delivers by default a number of FFh bytes equal to the value of Le (the expected data length). A different reply can be constructed if required by changing the contents of S$ as required. The status bytes SW1 and SW2 can also be changed from their default values of 90 00 depending on the desired effect on the reader.

Terminal application

Having collected some information in the file Card.Log we will want to read it from the card for further analysis. The file is normally left open so that data from several consecutive sessions can be logged, and so the first thing the program RFIDutil.BAS has to do is send the command C8 04 00 00 00, which closes the file as follows:

```
Command &HC8 &H04 COP(Lc=0,S$)
Close#1
S$="(c)2009 Patrick GUEULLE"
End Command
```

Just two corresponding lines are required in the source code for the terminal:

```
Declare Command &HC8 &H04 COP(S$,Le=&H17)
Call COP(S$)
```

Recovering the contents of the file is equally straightforward. In the terminal code we add the prefix '@:' to the filename, and read the file as normal. The operating system generates all the necessary commands automatically:

```
Open"@:card.log" For Input As #1
```

The following instruction is then used to extract one by one the commands for which the file contains the reply information:

```
Input#1, Z$
```

The rest of the terminal code is concerned with converting the contents of the data file into readable text, storing it on the hard disk and displaying it on the screen.

A practical example

Once you have had a look at the manual you can decide whether you prefer to use the development environment, which is well suited to organising projects, or to drive the ZCMBasic compiler from the command line. The result of compilation is a file RFIDutil.EXE, which can be run directly from the Windows command line, and a file RFIDspy.IMG (or RFIDspy.DBG), which has to be loaded into the memory on the card. With the card thus prepared, all you need to do is bring it within range of the terminal whose characteristics you are investigating.

The author tested the card at the point-of-sale terminal at the checkout in a French supermarket. The supermarket accepts payments of up to 20 Euro using contactless EMV cards such as MasterCard PayPass or Visa payWave. The BasicCard was brought within range of the terminal immediately before the real payment was carried out using a conventional bank card. Subsequent analysis of the file Card.Log revealed a sequence of select commands resembling the following:

```
00 A4 04 00 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31
00 A4 04 00 07 A0 00 00 00 04 30 60
00 A4 04 00 07 A0 00 00 00 04 10 10
00 A4 04 00 07 A0 00 00 00 04 99 99
00 A4 04 00 07 A0 00 00 00 03 20 10
00 A4 04 00 07 A0 00 00 00 03 10 10
00 A4 04 00 07 A0 00 00 00 43 10 10
00 A4 04 00 07 A0 00 00 00 42 10 10
```

The initial 00 stands for the ISO class (CLA) of the command, and A4 for the opcode (INS). Then 04 00 give the parameters P1P2, followed by a length indicator byte (Lc) and the application identifier (AID). We have discarded the null byte at the end of each line as it is not of any interest for analysis and only serves to indicate that the command does not expect a reply (Le = 0).

The first line represents an attempt to select the 'PPSE' (Proximity Payment System Environment) with the identifier, transmitted in ASCII, 2PAY.SYS.DDF01. This is exactly analogous to PSE in the case of EMV cards with contacts, which use the identifier 1PAY.SYS.DDF01 [5]. Our card replies with invalid data and so the terminal deduces that the PPSE, which at this point would normally supply a list of applications supported by the card, is not available. The terminal then proceeds to attempt to select in turn all the applications which it supports in the hope of finding one which also recognised by the card.

The next two lines show the terminal attempting to select two MasterCard applications, with priority given to Maestro (A0 00 00 00 04 30 60), a card which requires systematic (online) authorisation.

Before the terminal goes on to attempt to select the Visa Electron (A0 00 00 00 03 20 10) and Visa credit/debit (A0 00 00 00 03 10 10) applications, there is an attempt to select the mysterious application A0 00 00 00 04 99 99. This might correspond to a supermarket loyalty card: the contactless reader is apparently capable of dealing with these as well as with bank cards. The final selection attempt is for the French Carte Bleue credit card (A0 00 00 00 42 10 10).

What's in the kit?

The BasicCard is developed by ZeitControl, a small business based in Germany that evolved from being a vendor of time tracking systems into a specialist in chip cards. The first BasicCard was produced in 1996.

Compared to earlier BasicCard kits the 'dual interface' version features the addition of an RFID prototyping board carrying the TagTracer 14443 with a USB connection, buzzer, LED indicators and a printed antenna. These make developing applications much easier. In the author's opinion this development kit is a distinctive product, and is considerably more flexible than other similar units on the market.

- Omnikey 5321 USB – dual interface PC/SC smart card reader/writer
- Pocket card reader (balance checker)
- Development PCB for contactless ISO 14443 USB reader/writer
- Software development kit (SDK) for Windows
- Documentation on CD-ROM
- Printed technical manual (250 pages)
- 4 off BasicCard ZC7.5 Combi (32 kbyte EEPROM)

Further information is available at www.basiccard.com

On the other hand, a terminal that only accepts cards with contacts (such as a public telephone or petrol pump) might attempt to select applications in the following sequence:

```
00 A4 04 00 07 A0 00 00 00 42 10 10
00 A4 04 00 07 A0 00 00 00 42 20 10
00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31
00 A4 04 00 07 A0 00 00 00 03 10 10
00 A4 04 00 07 A0 00 00 00 03 20 10
00 A4 04 00 07 A0 00 00 00 04 10 10
00 A4 04 00 07 A0 00 00 00 04 30 60
```

Here we see that the terminal attempts to select French bank cards before the PSE. Only after that does it attempt to select international applications. In both cases the terminal's strategy is designed to make the transaction as quick as possible, which is especially critical in contactless applications.

One will sometimes encounter an attempt to select the Moneo application (`00 A4 04 00 06 A0 00 00 00 69 00`): this is an electronic wallet that is available in versions with and without contacts. Identifiers more than ten bytes long betray the existence of a card 'co-branded' with one or more commercial suppliers.

The author's next step is to experiment with contactless bank cards outside his home country. Contactless payment systems are being rolled out in many European countries including the UK, and it is expected that the system will be in widespread use for small payments within the next few years.

(090378)

Internet Links

[1] www.basiccard.com

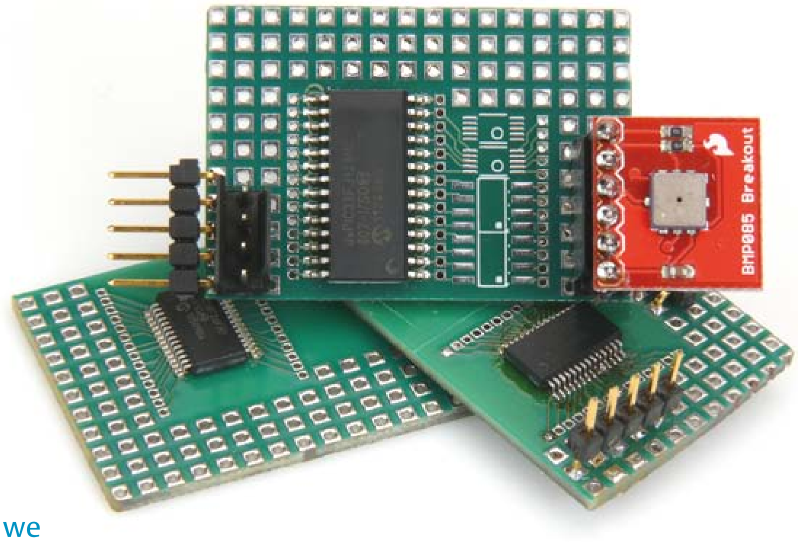
[2] www.elektor.com/090378

[3] www.elektor.com/010138

[4] www.emvco.com

PICo PROto

Minimalist prototyping tool for PIC16 or 18



By Michel Kuenemann (France)

How can you set about quickly testing a new detector or evaluating a new idea without having to modify or wire up a complex, expensive development board? I ask myself this question at least twice a day in busy periods — how about you? Working on the principle that the simplest things are the ones we use the most, I had the idea for the PICo PROto, a minimalist prototyping tool.

Main technical specifications

- accepts SO28 and SSOP28 packages
- minimal size and cost
- maximum flexibility
- lightning-fast implementation

Bulimic for new technologies, I like above all to evaluate new devices for myself, especially in the area of MEMS (microelectromechanical systems) detectors. Usually powered from 3.3 V, they deliver their data on analogue outputs or via an I²C bus, or less commonly on an SPI bus. Many of these modern detectors are encapsulated in tiny packages that are impossible to solder manually. Fortunately, the manufacturers of these marvels do think about the developers who have to implement them, and often offer a small evaluation board with the detector soldered onto it. They call these 'break-out boards' (BOB), they are fitted with a 0.1 in. (2.54 mm) pitch connector connected wire-for-wire to the main device.

I also regularly use several types of microcontrollers from the Microchip PIC18 family in 28-pin packages. As Microchip has had the very good idea of standardizing the pinout for these devices, it's easy to go from one type of device to another. What's more, as these devices also exist in traditional through-hole DIP packages, it's perfectly on the cards to build test set-ups on experimentation boards without any soldering at all. However, these spring-contact boards do have three major drawbacks: firstly, their size and weight are not compatible with my onboard applications in the field of model aircraft. Secondly, their electromechanical contacts are pretty random, not to say chaotic; and thirdly, you can't keep these set-ups for long, as we need to free up the precious board in double quick time for a new experiment.

You'd be right to point out that Microchip does offer many inexpensive evaluation boards that are just waiting to be adapted to my ideas, but here again, for most of my tests, the wealth of connectors on these boards, their size, and the resulting price are out of all proportion to my actual needs.

So I've designed a minimalist double-sided board for a PIC16 or 18, in an SO28 package on one side or an SSOP28 on the other, along with a few additional SMD components. Around the part occupied by the SMD components, there's an area with square pads on a 2.54 mm pitch that can take through-hole components, and the inevitable pin headers for connecting the prototype to the outside world. It's small size – only 28 × 42 mm – means it can be built just as well into a scale model aircraft or a small mobile robot as into a more roomy project. The good thing about the small size constraint is that it forces the constructor to stay focused on the objective

This pinout is suitable for all PIC16 or 18 devices in a 28-pin package

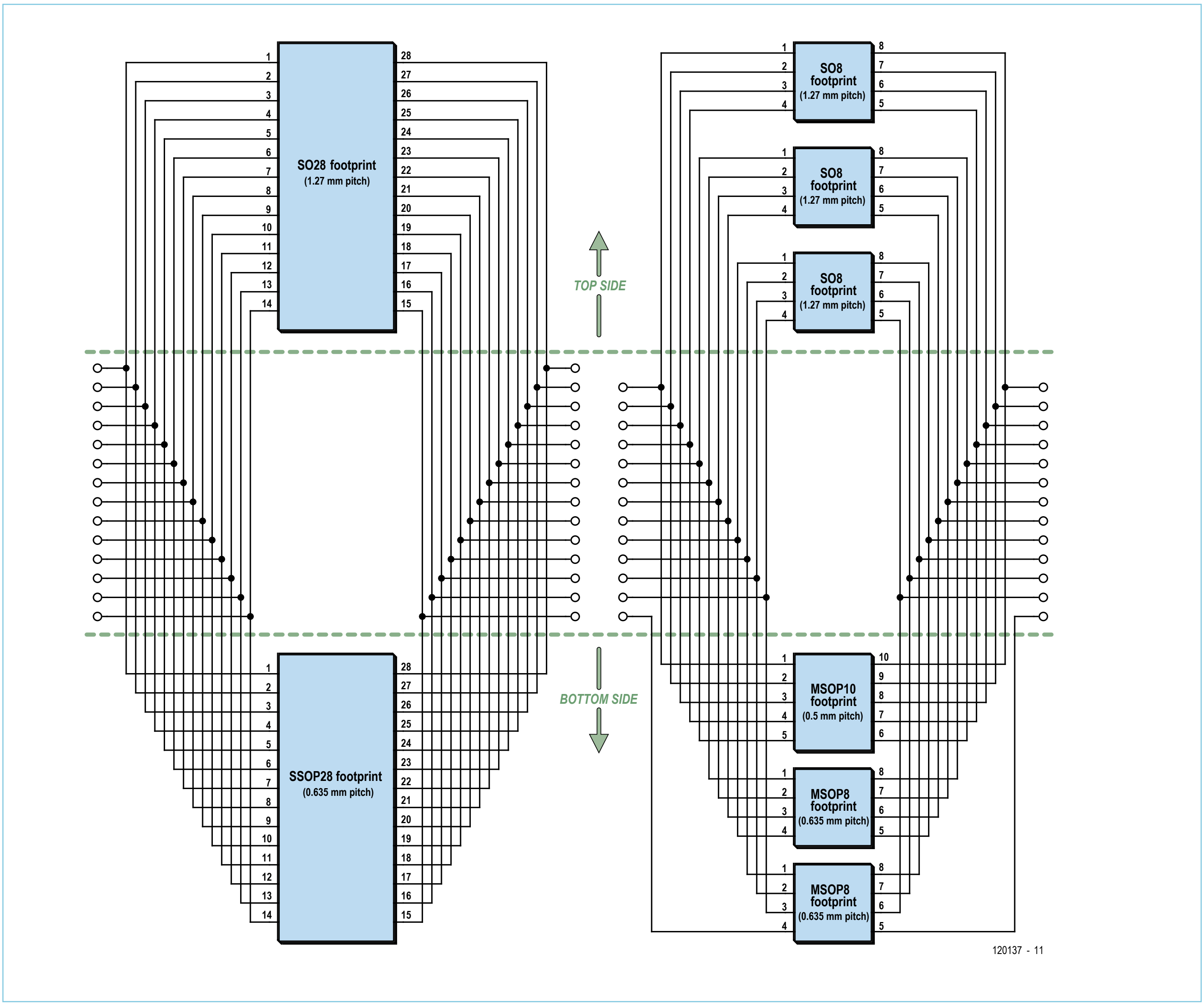


Figure 1. This is not a circuit diagram in the usual sense, but more of a phantom, as it doesn't show the components, just their footprints. The PICo PROto is compatible with all 8-bit PICs in 28-pin packages.

of their operation, without giving in to the temptation to "weigh down the donkey" with superfluous components.

Circuit

No surprise here, it's transparency itself; besides, we can hardly call it a circuit (**Figure 1**) as there are no components. So we can move straight on to the practical side. In addition to the SO28 and SSOP28 footprints for the controller (only fit one at a time, please!), I have allowed for SO8 and MSOP8 footprints too. This can be seen more clearly still in the enlarged reproduction of the double-sided PCB (**Figure 2**). The photos show the top and underside of two of my **PICo PROto**s (**Figure 3**). Their layout is slightly different from the final version. On the left, without components, on the right the top side with a TSSOP28 (the quality of the soldering is far from exemplary).

How to use PICo PROto?

Start by soldering the microcontroller you've chosen onto the top or underside, depending on the type of package. Then at the edge of the board solder in a 5-pin header (2.54 mm pitch) – using a right-angle type if necessary. This will let you

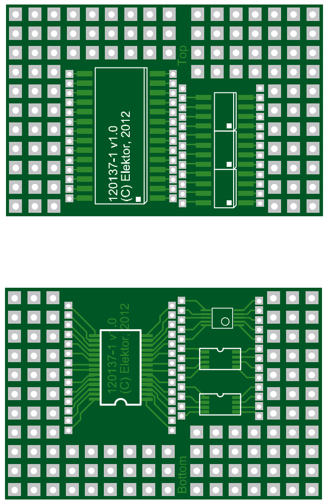


Figure 2. The through-hole plated double-sided PICo PROto PCB at a scale of 150%.

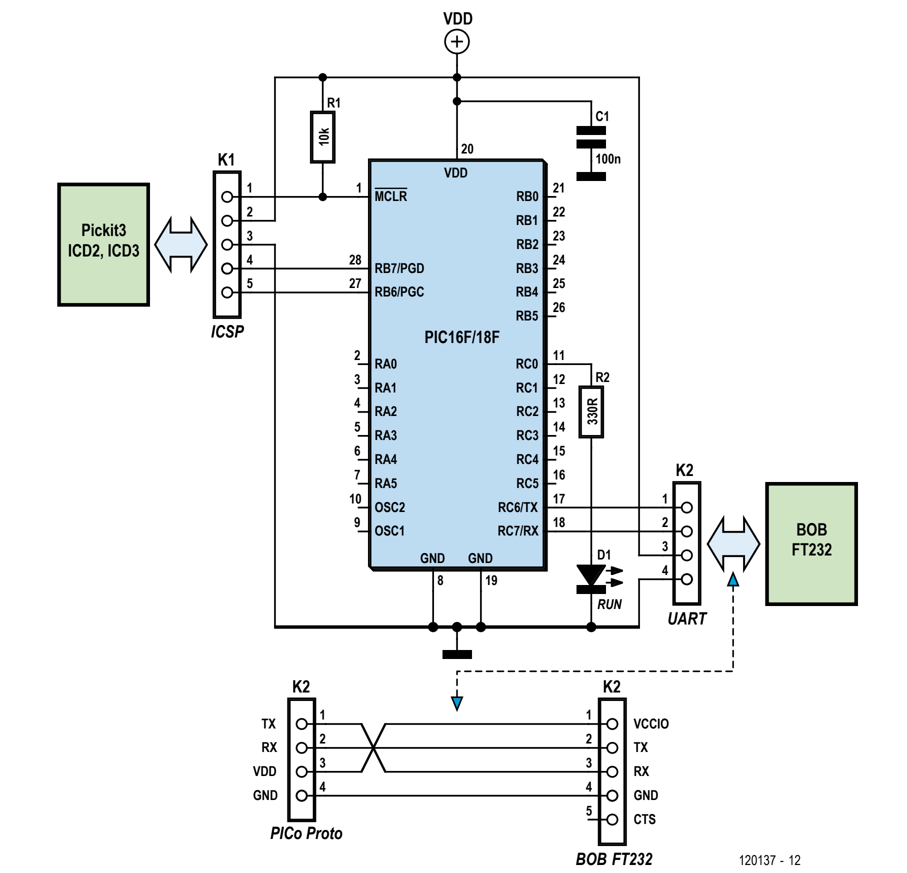


Figure 3. Wiring up the PICo PROto.

connect the vital PICkit or ICD programmer for flashing and debugging your programs. Connect the header to the microcontroller as shown in the diagram (**Figure 3**), but don't go any further for the moment. Check that your MPLAB development environment sees and recognizes the device. To do this, connect your programmer to the board and to your development PC. Run MPLAB, then in the 'Programmer' menu, select your tool and check that it connects correctly. Let's assume you are using a PICkit3. At this point, MPLAB will return the following error: "You must connect to a target device to use PICkit". This error is due to the fact that your microcontroller is not yet powered up. Go into Programmer → Settings → Power, check that the cursor is on the voltage suitable for your application and check the "Power target circuit from PICkit3" box. In Configure → Select device, select the exact type of microcontroller you are using, and everything should then be OK. The least exciting part of the adventure is now behind you!

If you want to communicate with your microcontroller using the USB interface, make use of a BOB FT232 module [2]. After soldering in and wiring up connector K2, you'll be able to make the connection; the BOB FT232 will also take care of powering your board (see the PICo PROto / BOB

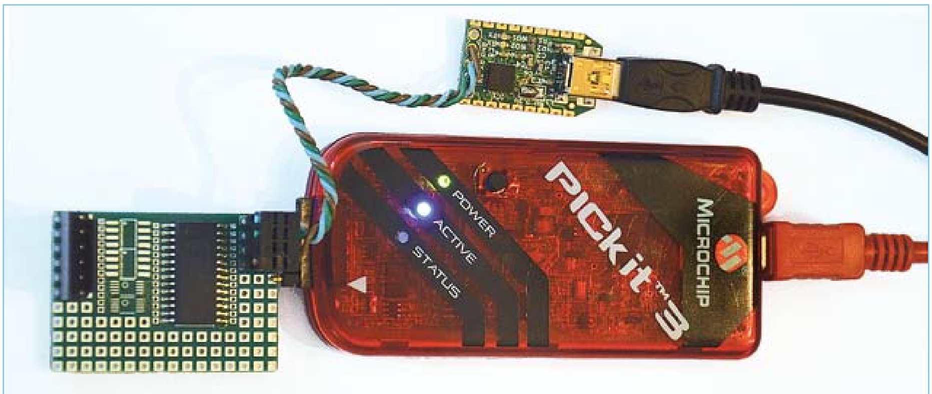


Figure4a. Application using the BOB for evaluating a barometric sensor.

interconnection diagram at the bottom of Figure 4). In this case, before connecting up the BOB, don't forget to uncheck the "Power target circuit from PICkit3" box in the MPLAB. On the BOB, you will also have taken care to select the right supply voltage for your circuit using jumper JP1.

If you find it reassuring to see a red LED flashing to tell you the current program is running OK, don't be afraid to fit a red LED as shown in the diagram.
A seasoned experimenter will need less than an hour to get to this point in the project. The free space is now available for your boldest testing.

To end with, here are two more examples (**Figure 4**) of the **PICo PROto** in use:

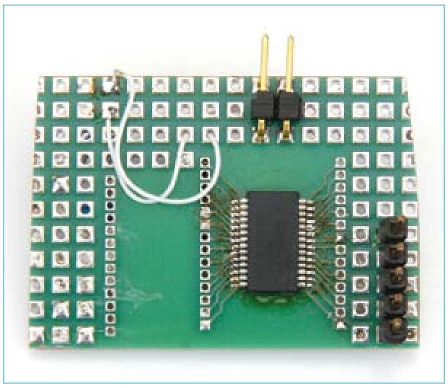


Figure 4b. Underside: PIC18F27J13 in an SSOP28 package.

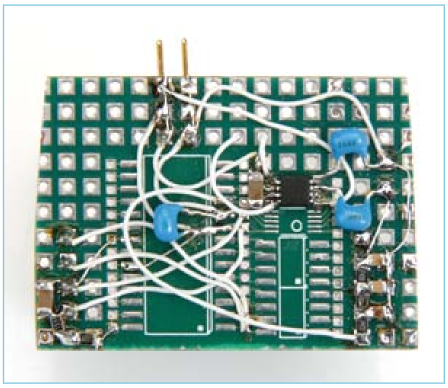


Figure 4c. Top side, an SSM2301 Class D amplifier being tested for the lengths counter for swimmers published in the Summer 2012 issue.

Evaluating a barometric sensor, using the BOB
Evaluating a voice synthesis process with a PIC18F27J13 in an SSOP28 package on the underside and an SSM2301 Class D amplifier on the top.

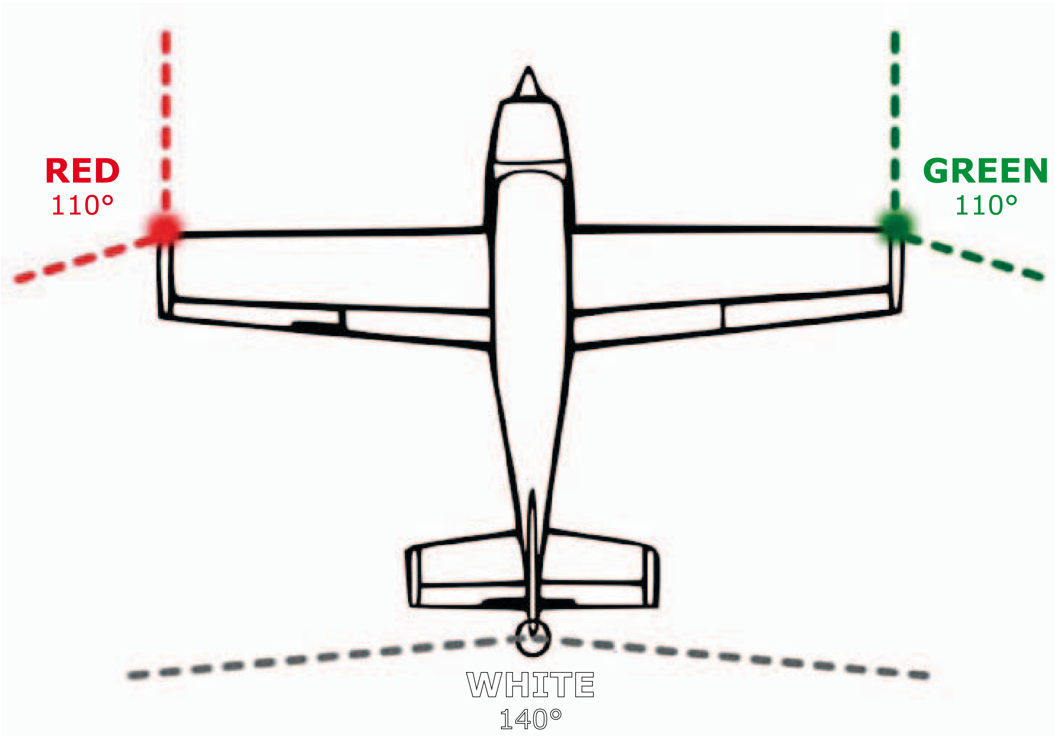
120137-I

Internet Links

[1] www.elektor.com/120137

[2] BOB-FT232R USB/serial Bridge, Elektor September 2011, www.elektor.com/110553

Model Aircraft Lighting

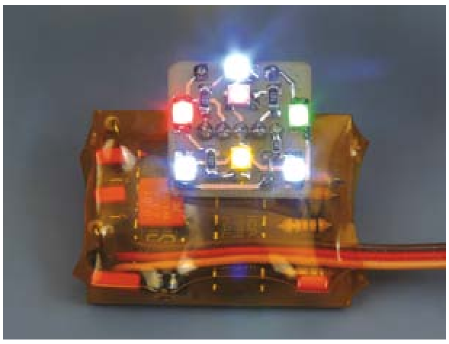


This design should gladden the heart of any model aircraft enthusiasts: it is a circuit to control the navigation lights, landing light, anti collision beacon and wing tip strobes from a spare channel of a model remote controller. The lights are switched remotely using any spare servo channel on the remote control receiver.

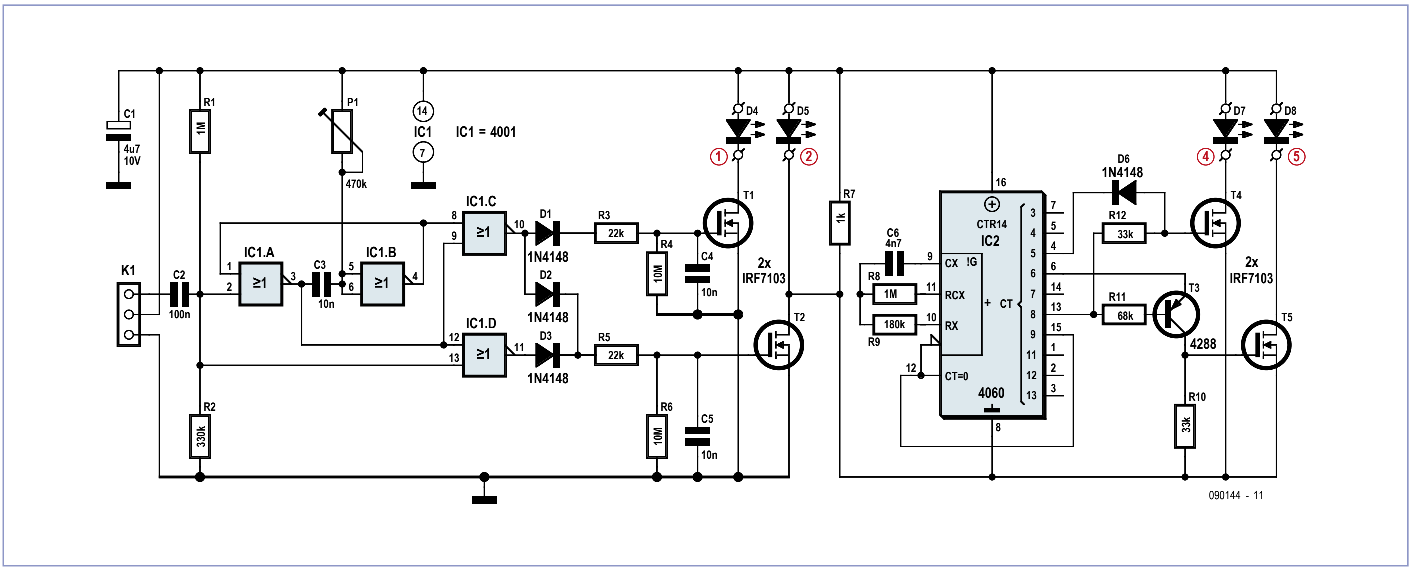
By Werner Ludwig (Germany)

Model servos are controlled by a PWM signal. The pulse width determines the position of the servo output arm. A 1 ms pulse will drive the arm to one end of its travel (viewed from above servo fully anticlockwise) and a 2 ms pulse drives it to the other end (fully clockwise).

The servo control pulse output signal from the receiver plugs into connector K1. The circuit consists of a pulse-width discriminator circuit which controls the lights depending on the length of the pulses from the transmitter. The pulse-width discriminator is constructed using four NOR gates. IC1.A and IC1.B are configured to form a monostable flip flop which produces an output 'reference' pulse width of 1.5 ms. This pulse width corresponds to the servo neutral position. When the input pulse width is either longer or shorter than this reference pulse it will cause an output pulse from either IC1.C (input pulse longer than reference pulse turns on landing light plus all other lights)



or from IC1.D (input pulse shorter than the reference pulse turns on all lights except landing light). In the quiescent condition IC1.C and IC1.D will both have a zero output state. The resistor/capacitor networks on the outputs of IC1.C and IC1.D smooth out the positive pulses to produce a stable high level which switches either transistor T1 or T2 on. These driver transistors will then switch the load. The transistors used in the author's prototype version are capable of switching 2 A, making them suitable for driving low voltage filament lamps as well as LEDs. T1 switches the aircraft landing light, which is a high efficient white power LED. Diode D1 at the output of IC1.C performs a logic 'OR' function ensuring that when the landing light is on all of the other navigation lights will also be on. These are shown in the first circuit as the navigation lights (left = red, right = green and tail light = white) the white wing tip strobes and the anti-collision light (ACL beacon). The navigation lights are switched by transistor T2 and the other branch from T2 switches an earth return to the pulse generator stage to flash the strobe and ACL beacon. The pulse generator consists of IC2 (4060) which is a 14-stage binary counter with built-in clock generator. The circuitry on the output of



this chip provides a four flash output for the strobe light and a double flash output for the ACL beacon.

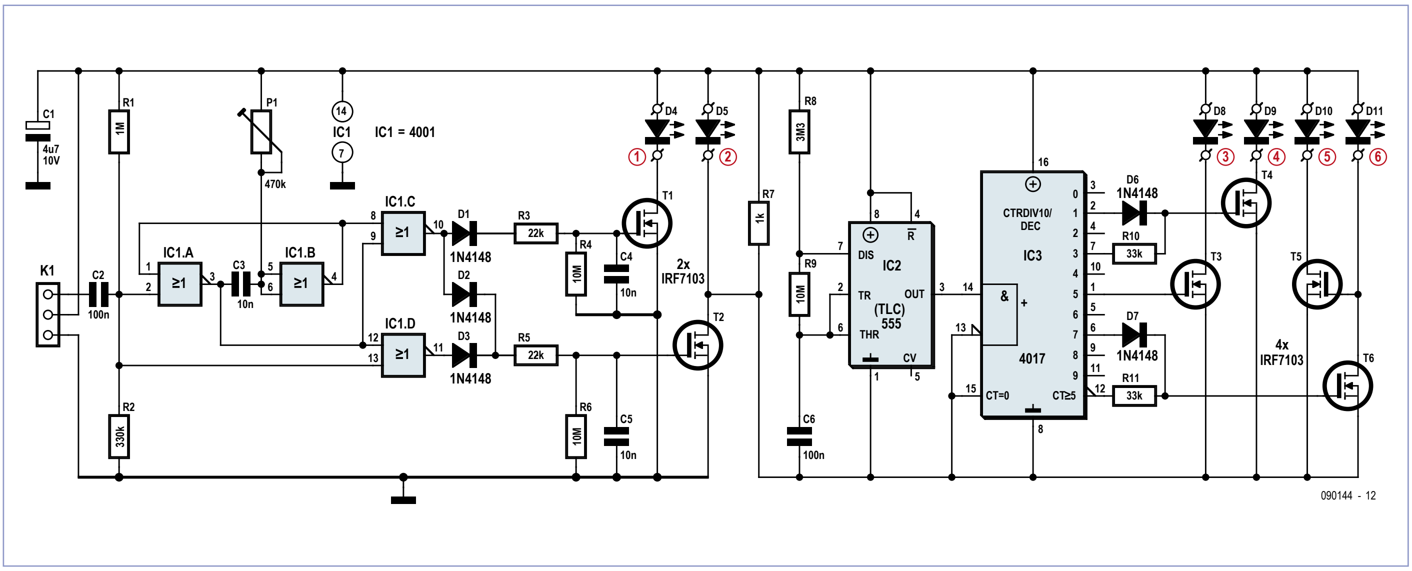
An alternative version of the aircraft light controller is also included here. This one uses a 10-stage decimal counter type 4017 to provide control signals to switch the lights. This type of IC does not have an internal oscillator so a NE555 timer is included to provide the clock signal. The counter outputs uses transistor logic to gate the correct pulses to the LEDs. In contrast to the first circuit this one provides separate signals for wing tip rear and forward strobes and tail ACL beacon.

One final tip: LEDs are ideal for this sort of application, compared to filament lamps they are physically smaller and much more robust, using little energy and have a long life expectancy. They also have quite a narrow beam width which in this application is a disadvantage. This can however be remedied by lightly abrading the LED front surface and then painting with clear varnish to give a more even light dispersion.

(090144)

Table 1: Circuit diagram lamp designation

1	Landing light
2	Navigation lights
3	Rear wing tip strobes
4	Forward wing tip strobes
5	ACL beacon
6	Tail strobe



The 'Pansanitor' (1928)

By Dipl-Inf Karl-Ludwig Butte (Germany)

1

2

Electrical muscle stimulation machines offer relief from aches and pains, muscle tension, neuralgia and lots of other miscellaneous ailments, using either low- or high-frequency alternating currents. Generating these currents is bread-and-butter stuff using modern semiconductor technology, but how was it done eighty years ago, in a time when transistors, let alone integrated circuits and microcontrollers, were not exactly off-the-shelf components? In this 'Retronics' article we take a rare opportunity to examine the practically forgotten know-how from that period, with the help of an original example of the 'Pansanitor', a muscle stimulation device dating from 1928.

Muscle stimulation machines: use and operation

"The radiation from high-voltage and high-frequency currents is, like the high voltages delivered by the electrophorus, believed to have a soothing effect on the nervous system in general and on sore nerves in particular." These words appear in the book *High frequencies for the sick and the healthy: a medical companion* [1], quoting Prof Dr Opitz from his *Gynaecology Handbook* [2].

However, that was not the only application of electrotherapy. Whether machines like the Pansanitor were used to treat the metabolism, respiratory organs, or skin and hair, the patient was soon restored to health (or at least that is what the books' publishers would have you believe).

Muscle stimulation machines are still used in medicine today, for example in cases of myaesthenia and circulatory disorders, and in general pain management. Thermal radiation is used in such cases, which has a deeper penetration than for example infra-red light.

The historical context

The device we are examining dates from 1928, the year in which Amelia Earhart becomes the first woman to cross the Atlantic, Alexander Fleming discovers penicillin, and Coca-Cola becomes the first commercial sponsor of an Olympic Games. Just three years earlier the physicist Julius Edgar Lilienfeld had filed the first patent covering the principle of operation of the transistor [3].

The basic technology that paved the way for the boom in muscle stimulation devices was developed by Nicola Tesla in the last years of the nineteenth century. In 1898 he published his research results in *The Electrical Engineer* under the title 'High frequency oscillators for electro-therapeutic and other purposes' [4]. Starting from an ordinary spark coil, he described, more from the point of view of an electrical engineer than that of a doctor, how his experimental devices were constructed and then refined. A spark coil consists simply of a transformer, a capacitor, a 'hammer' interrupter and a spark gap. For reasons of economy Tesla wired two capacitors in parallel to reduce the potential difference between the secondary-side connections and so save on materials. This had the disadvantage of reducing the frequency of the alternating current, which he compensated for by adding a so-called Tesla coil (see figure 4 in [4]). For electrodes Tesla used evacuated glass bulbs, subsequently filled with a gas that glowed violet while the machine was in operation. From this the device came to be known as the 'violet wand'. The shape of the glass bulbs was modified to adapt the device to the particular complaint or bodily part being treated: a selection can be seen in Figure 1.

Retronics is a monthly section in Elektor magazine covering vintage electronics including legendary Elektor projects. Contributions, suggestions and requests are welcomed; please send an email to Jan Buiting (editor@elektor.com).

Electrical muscle stimulation from the olden days

Description of the Pansanitor

The Pansanitor unit, along with all its accessories including cables, electrode holders and its wide array of glass electrodes, came in a handy carrying-case with a plush violet lining (**Figure 1**). The control unit for generating the stimulation currents was built into a black wooden housing, with a plastic ivory-coloured top panel carrying the legends for the two rotary controls and for the connection sockets (**Figure 2**). The device's logo also appears in large letters, of course, along with the model number. It is interesting to note that there is no indication as to the manufacturer of the device: I have searched the internet for such information, but so far without success.

On the left-hand side are four pins in a T-shaped arrangement, which were used to connect the device to the mains. The middle pin on the left is the neutral connection, while the other three allow for operation from 110 VAC, 150 VAC or 220 VAC. We therefore know that the manufacturer had an eye to the export market, intending the unit to be used with different line voltages. It behoved the customer to select the right pin for his AC voltage as there was no overvoltage protection! Indeed, the unit is not even fused. Black plastic caps were fitted over the unused pins.
The AC power cable on this example has been replaced by the previous owner, but the original porcelain plugs are still present and in good condition (**Figure 3**).

Of the two large black rotary controls the one on the left functions as an on-off switch while the one on the right controls the intensity of the device's output.
The glass electrodes are connected to the two central sockets on the device using the cables on the electrode holder, which also acts as a handle (see **Figure 4**). However, out of concern for my own safety I have not dared to switch the machine on!

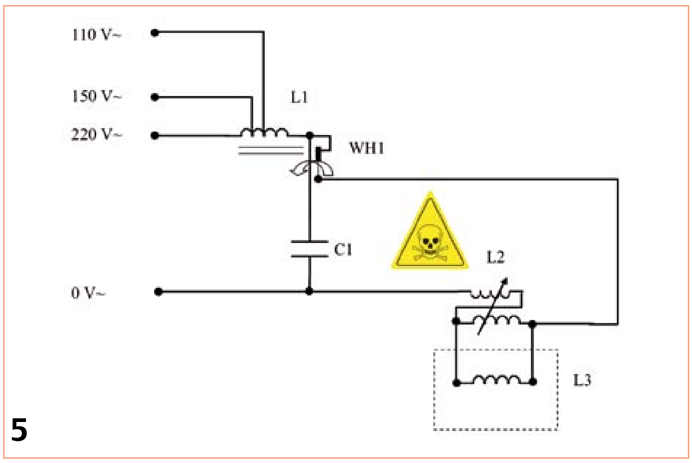
Circuit of the Pansanitor

Figure 5 shows the circuit diagram of the Pansanitor and **Figure 6**, the insides of the machine. Coil L1 has two extra taps, to allow for the various supply voltage options. This means that the whole device is at mains potential! L1, together with the mechanical contact WH1, forms the hammer interrupter; in conjunction with a cam on the On-Off switch, this also controls overall power to the unit (**Figure 7**). When the main switch is turned and power is applied, a magnetic field builds up in L1 which attracts the switching contact WH1. As a result the electrical circuit is broken and the large back EMF from the coil begins to charge C1. C1 has a capacitance of 25 nF and is rated (or perhaps it would be better to say 'was rated') for operation at up to 2 kV. Given the antiquity of the component it is probably best not to rely on that figure anymore!

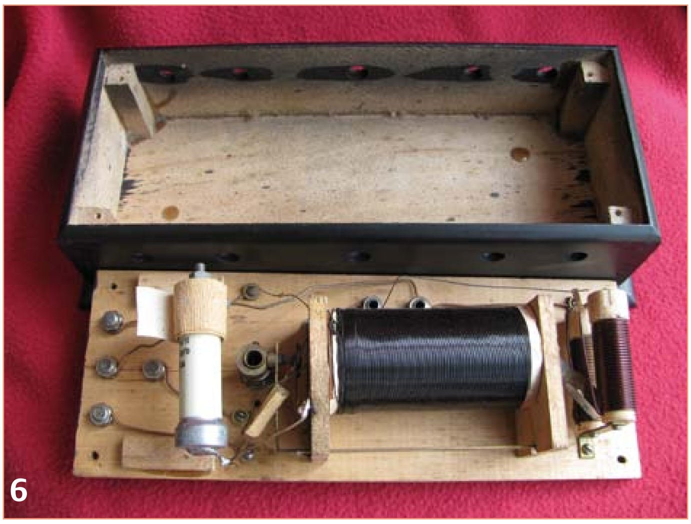
With the supply voltage interrupted the switching contact WH1 returns to its original position, causing current to flow and the whole process to start again from the beginning.

3

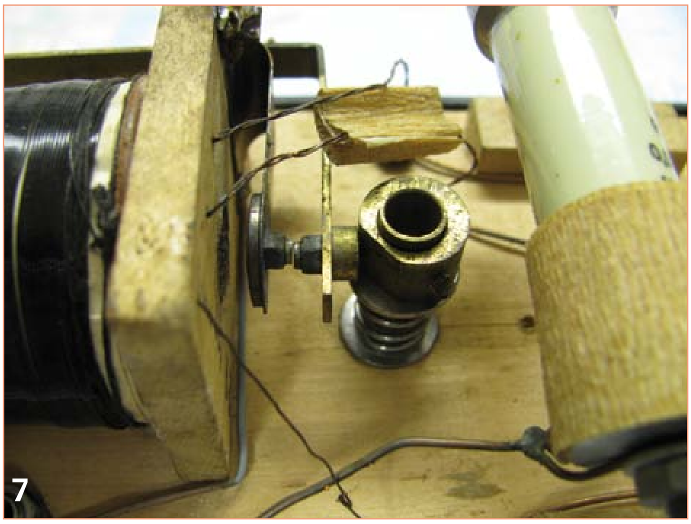
4



5



6



7

The high voltage thus produced is then converted to an even higher voltage by the Tesla transformer comprising a primary coil L2 and a secondary coil L3. L2 consists of two series-connected windings mounted one above the other (**Figure 8**). A pair of sprung contacts between the two windings, which can be moved using the intensity control knob, gives a certain amount of control over the final voltage produced: the contacts simply act as adjustable taps on the windings, much like a wirewound potentiometer.

The secondary coil L3 (Figure 4) is located in the handheld part of the device into which the various glass electrodes are fitted, and is connected to the main unit using a plug-and-socket arrangement. L3 has just ten turns using doubled-up conductors (**Figure 9**).
The whole circuit is mounted on the back of the lid of the unit. In terms of simplicity and elegance it is hard to beat.

Tesla transformer included!

Just as the microcontroller is the most interesting component in the modern version of the circuit, so the Tesla coil commands centre stage in the 1928 Pansanitor. Microcontrollers are now part of everyday life, and many Elektor readers will be on friendly terms with them. The Tesla coil, on the other hand, now holds a certain mystique that will attract any electronics fan. That must originate not just from the charismatic personality of its inventor, Nikola Tesla, but also from its best-known incarnation, Wardenclyffe Tower (also called the 'Tesla tower') [5], which was sadly demolished in 1917. Pictures of enormous electrical discharges can easily be found [6]. The chief fascination of the Tesla transformer, however, is the fact that it is not just the turns ratio between the primary and secondary windings that is responsible for determining the magnitude of the voltage produced, but that resonance between the two coils also plays an important role [7].

(120177)

References

[1] Thüringer Verlagsanstalt und Druckerei GmbH: Jena, 1930; www.electrotherapymuseum.com/2005/HF/index.htm

[2] Bergmann: Munich, 1922; page 354

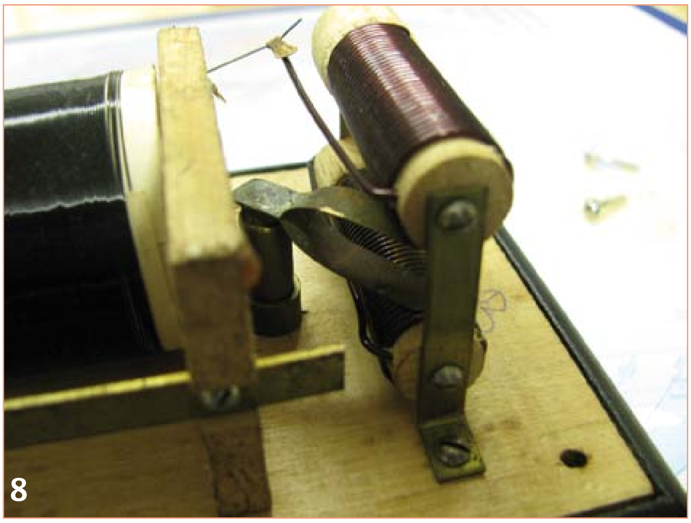
[3] http://en.wikipedia.org/wiki/Transistor

[4] The Electrical Engineer, volume XXVI, number 550, November 17 1898; www.tfcbooks.com/tesla/1898-11-17.htm

[5] http://en.wikipedia.org/wiki/Wardenclyffe_Tower

[6] http://en.wikipedia.org/wiki/File:Lightning_simulator_questacon02.jpg

[7] http://en.wikipedia.org/wiki/Tesla_coil



8

9

Hexadoku

Puzzle with an electronics touch

If you have the odd minute to spare after reading and doing all that electronics wizardry in this edition, try solving this Hexadoku served fresh for your amusement. Enter the right numbers or letters A-F in the open boxes, find the solution in the grey boxes, send it to us and you automatically enter the prize draw for one of four Elektor Shop vouchers.

The instructions for this puzzle are straightforward. Fully geared to electronics fans and programmers, the Hexadoku puzzle employs the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation. Correct entries received enter a draw for a main prize and three lesser prizes. All you need to do is send us the numbers in the grey boxes.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for one Elektor Shop voucher worth £80.00 and three Elektor Shop Vouchers worth £40.00 each, which should encourage all Elektor readers to participate.

Participate!

Before October 1, 2012,
send your solution (the numbers in the grey boxes) by email to:

hexadoku@elektor.com

Prize winners

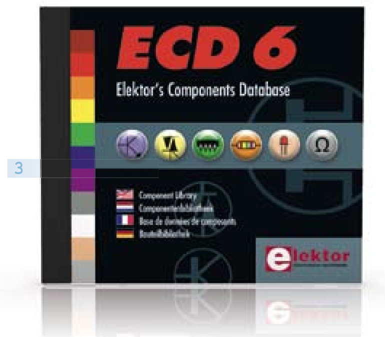
The solution of the June 2012 Hexadoku is: **7924A**.
The Elektor £80.00 voucher has been awarded to Thomas Raith (Germany).
The Elektor £40.00 vouchers have been awarded to Reino Anttila (Finland), Michael Evans (UK), and Nuno Tavares (Portugal.
Congratulations everyone!

		C		9							E		6		
7			B	C	0					A	F	3			5
A	6					4			0					7	C
E	F			2		8			3		D			A	B
		3		B	2		F	4		E	1		A		
			F	3	4	0			6	B	A	E			
2	5		D		7	A			9	0		8		B	4
	A													9	
	8													0	
F	2		9		A	C			B	8		6		E	D
			E	5	8	2			A	4	0	F			
		5		F	1		6	9		D	C		2		
5	C			8		6			F		4			2	E
B	0					3			2					8	1
8			2	A	E					5	7	C			0
		4		7							8		5		

7	8	C	5	B	F	0	1	E	9	3	6	2	D	4	A
3	A	E	0	7	9	2	8	C	D	B	4	5	6	F	1
4	D	9	B	5	6	C	E	A	2	F	1	0	3	7	8
1	F	2	6	A	D	3	4	0	5	7	8	E	C	9	B
5	C	6	1	F	A	D	0	2	7	8	9	3	4	B	E
8	0	D	4	C	B	1	5	3	E	6	F	9	A	2	7
9	2	A	7	E	3	4	6	B	C	1	5	F	0	8	D
B	E	F	3	8	7	9	2	4	A	D	0	C	1	6	5
E	4	0	8	2	C	F	A	5	B	9	D	6	7	1	3
A	3	5	2	D	E	6	B	F	1	0	7	4	8	C	9
C	7	1	F	0	4	8	9	6	3	A	E	D	B	5	2
D	6	B	9	1	5	7	3	8	4	C	2	A	E	0	F
F	5	8	C	9	0	A	D	7	6	E	B	1	2	3	4
2	1	4	E	3	8	B	C	9	0	5	A	7	F	D	6
6	B	3	A	4	1	5	7	D	F	2	C	8	9	E	0
0	9	7	D	6	2	E	F	1	8	4	3	B	5	A	C

1


2


3


LabWorX 2

1 Mastering Surface Mounted Technology

This book takes you on a crash course in techniques , tips and know-how to successfully introduce Surface Mounted Technology in your workflow. Even if you are on a budget you too can jumpstart your designs with advanced fine pitch parts. Besides explaining methodology and equipment, attention is given to parts technology and soldering technique. Several projects introduce you step by step to handling surface mounted parts and the required technique to successfully build SMT assemblies. Many practical tips and tricks are disclosed that bring surface mounted technology into everyone's reach without breaking the bank.

Approx. 260 pages • ISBN 978-1-907920-12-7 • £29.50 • US $47.60

A whole year of Elektor magazine onto a single disk

2 DVD Elektor 2011

The year volume DVD/CD-ROMs are among the most popular items in Elektor's product range. This DVD-ROM contains all editorial articles published in Volume 2011 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favourite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-276-1 • £23.50 • US $37.90

More than 70,000 components

3 CD Elektor's Components Database 6

This CD-ROM gives you easy access to design data for over 7,800 ICs, more than 35,600 transistors, FETs, thyristors and triacs, just under 25,000 diodes and 1,800 optocouplers. The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. All databank applications are fully interactive, allowing the user to add, edit and complete component data.

ISBN 978-90-5381-258-7 • £24.90 • US $40.20

Elektor Linux Board

4 Embedded Linux Made Easy

Today Linux can be found running on all sorts of devices, even coffee machines. Many electronics enthusiasts will be keen to use Linux as the basis of a new microcontroller project, but the apparent complexity of the operating system and the high price of development boards has been a hurdle. Here Elektor solves both these problems, with a beginners' course accompanied by a compact and inexpensive populated and tested circuit board. This board includes everything necessary for a modern embedded project: a USB interface, an SD card connection and various other expansion options. It is also easy to hook the board up to an Ethernet network.

Populated and tested Elektor Linux Board
Art.# 120026-91 • £57.80 • US $93.30

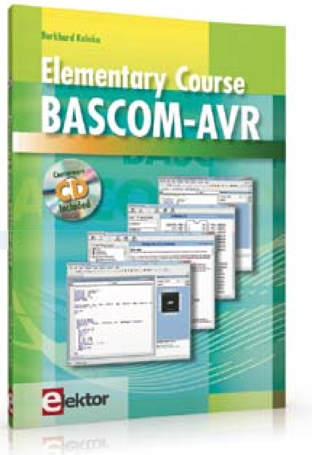
A comprehensive and practical how-to guide

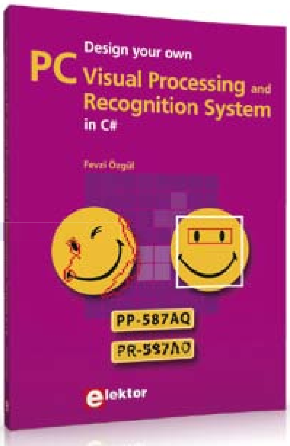
5 Design your own PC Visual Processing and Recognition System in C#

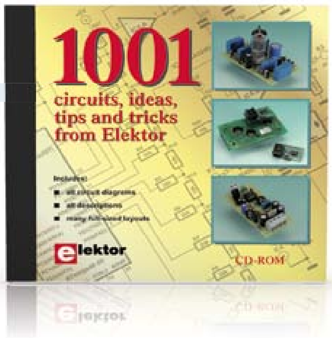
This book is aimed at Engineers, Scientists and enthusiasts with developed programming skills or with a strong interest in image processing technology on a PC. Written using Microsoft C# and utilizing object-oriented practices, this book is a comprehensive and practical how-to guide. The key focus is on modern image processing techniques with useful

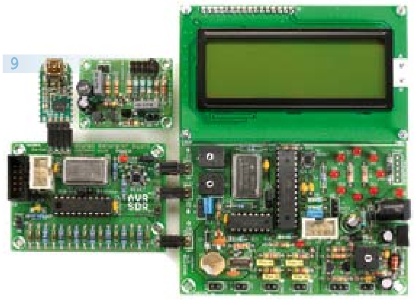
4


7


6


5


8


9


and practical application examples to produce high-quality image processing software. Analysis starts with a detailed review of the fundamentals of image processing. It progresses to explain and explore the prac tical uses of two highly sophisticated and freely downloadable, open source image processing libraries; AForge.NET and Emgu.CV, utilizing dotnet technology within the Microsoft Visual Studio environment. All code examples used are available – free of charge – from the Elektor website; you can easily create and develop your own results to demonstrate the concepts and techniques explained.

307 pages • ISBN 978-1-907920-09-7

£35.50 • US $57.30

Free Software CD-ROM included

6 Elementary Course BASCOM-AVR

The Atmel AVR family of microcontrollers are extremely versatile and widely used. In Elektor magazine we have already published many interesting applications employing an ATmega or ATtiny microcontroller. The majority of these projects perform a particular function. In this book we focus more on the software aspects. Using lots of practical examples we show how, using BASCOM, you can quickly get your own design ideas up and running in silicon.

224 pages • ISBN 978-1-907920-11-0

£34.95 • US $56.40

Ultrasensitive wideband E-smog detector

7 TAPIR Sniffs it Out!

Attention boy scouts, professionals and grandfathers! This ultrasensitive wideband E-smog detector offers you two extra senses to track down noise that's normally inaudible. TAPIR — short for Totally Archaic but Practical Interceptor of Radiation — also makes a nice project to build: the kit comprises everything you need. Even the enclosure, ingeniously consisting of the PCB proper! Using the TAPIR is dead easy. Connect the headphones and an antenna and switch it on. Move it around any electrical device and you'll hear different noises with each device, depending on the type and frequency of the emitted field.

Kit of parts, incl. PCB

Art.# 120354-71 • £13.30 • US $21.50

Circuits, ideas, tips and tricks

8 CD 1001 Circuits

This CD-ROM contains more than 1000 circuits, ideas, tips and tricks from the Summer Circuits issues 2001-2010 of Elektor, supplemented with various other small projects, including all circuit diagrams, descriptions, component lists and full-sized layouts. The articles are grouped alphabetically in nine different sections: audio & video, computer & microcontroller, hobby & modelling, home & garden, high frequency, power supply, robotics, test & measurement and of course a section miscellaneous for everything that didn't fit in one of the other sections.

ISBN 978-1-907920-06-6 • £34.50 • US $55.70

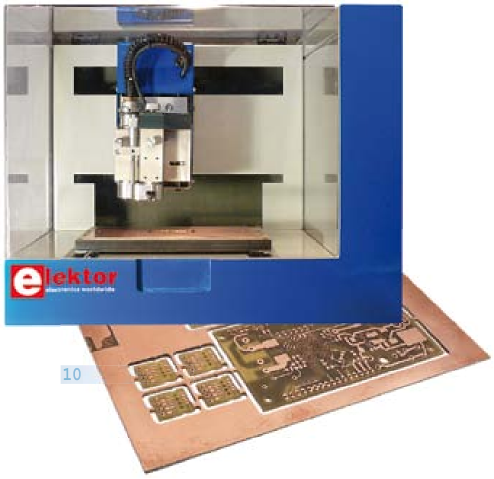
Package Deal: 12% off

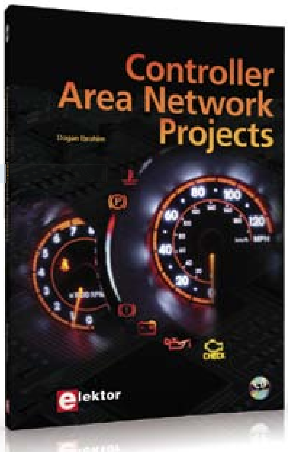
9 AVR Software Defined Radio

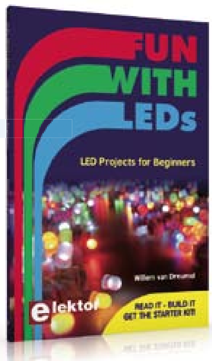
This package consists of the three boards associated with the AVR Software Defined Radio articles series in Elektor, which is built around practical experiments. The first board, which includes an ATtiny2313, a 20 MHz oscillator and an R-2R DAC, will be used to make a signal generator. The second board will fish signals out of the ether. It contains all the hardware needed to make a digital software-defi ned radio (SDR), with an RS-232 interface, an LCD panel, and a 20 MHz VCXO (voltage-controlled crystal oscillator), which can be locked to a reference signal. The third board provides an active ferrite antenna.

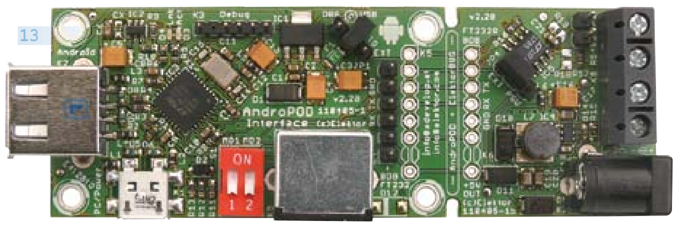
Signal Generator + Universal Receiver +Active Antenna: PCBs and all components + USB-FT232R breakout-board

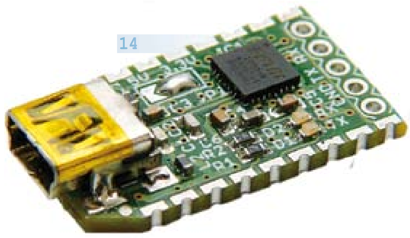
Art.# 100182-72 • £99.90 • US $133.00











A professional PCB router

10 Elektor PCB Prototyper

This compact, professional PCB router can produce complete PCBs quickly and very accurately. This makes the PCB Prototyper an ideal tool for independent developers, electronics labs and educational institutions that need to produce prototype circuits quickly. The PCB Prototyper puts an end to waiting for boards from a PCB fabricator –- you can make your own PCB the same day and get on with the job. Further information, demo video and ordering at www.elektor.com/prototyper.

Art.# 100619-71 • £3,100 • US $4,900

Free mikroC compiler CD-ROM included

11 Controller Area Network Projects

The aim of the book is to teach you the basic principles of CAN networks and in addition the development of microcontroller based projects using the CAN bus. You will learn how to design microcontroller based CAN bus nodes, build a CAN bus, develop high-level programs, and then exchange data in real-time over the bus. You will also learn how to build microcontroller hardware and interface it to LEDs, LCDs, and A/D converters.

260 pages • ISBN 978-1-907920-04-2

£29.50 • US $47.60

Associated 60-piece Starter Kit available

12 Fun with LEDs

This booklet presents more than twenty exciting projects covering LEDs, aimed at young & old. From an Air Writer, a Party Light, Running Lights, a LED Fader right up to a Christmas Tree. Use this book to replicate various projects and then put them into practice. To give you a head start each project is supported by a brief explanation, schematics and photos. In addition, the free support page on the Elektor website has a few inspiring video links available that elaborate on the projects. A couple of projects employ the popular Arduino microcontroller board that's graced by a galaxy of open source applications. The optional 60-piece Starter Kit available with this book is a great way to get circuits built up and tested on a breadboard, i.e. without soldering.

96 pages • ISBN 978-1-907920-05-9

£16.95 • US $38.00

Bridging Android and your electronics projects

13 AndroPod

With their high-resolution touchscreens, ample computing power, WLAN support and telephone functions, Android smartphones and tablets are ideal for use as control centres in your own projects. However, up to now it has been rather difficult to connect them to exter-nal circuitry. Our AndroPod interface board, which adds a serial TTL port and an RS485 port to the picture, changes this situation.

Andropod module with RS485 Extension

Art.# 110405-91 • £53.35 • US $74.70

Meet BOB

14 FT232R USB/ Serial Bridge/BOB

You'll be surprised first and foremost by the size of this USB/serial converter – no larger than the moulded plug on a USB cable! And you're also bound to appreciate that fact that it's practical, quick to implement, reusable, and multi-platform – and yet for all that, not too expensive! Maybe you don't think much of the various commercially-available FT232R-based modules. Too expensive, too bulky, badly designed, … That's why this project got designed in the form of a breakout board (BOB).

PCB, assembled and tested

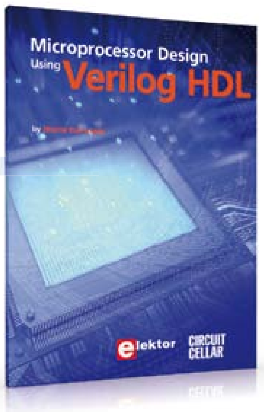
Art.# 110553-91 • £12.90 • US $20.90

A highly-practical guide

15 Linux – PC-based Measurement Electronics

If you want to learn how to quickly build Linux-based applications able to collect, process and display data on a PC from various analog and digital sensors, how to control circuitry attached to a computer, then even

18



17

19

15

16

how to pass data via a network or control your embedded system wirelessly and more – then this is the book for you! The book covers both hardware and software aspects of designing typical embedded systems using schematics, code listings and full descriptions.

264 pages • ISBN 978-1-907920-03-5

£29.50 • US $47.60

110 issues, more than 2,100 articles

16 DVD Elektor 1990 through 1999

This DVD-ROM contains the full range of 1990-1999 volumes (all 110 issues) of Elektor Electronics magazine (PDF). The more than 2,100 separate articles have been classified chronologically by their dates of publication (month/year), but are also listed alphabetically by topic. A comprehensive index enables you to search the entire DVD. What's more, this DVD also contains the entire 'The Elektor Datasheet Collection 1...5' CD-ROM series.

ISBN 978-0-905705-76-7 • £69.00 • US $111.30

Processor design in the real world

17 Microprocessor Design using Verilog HDL

If you have the right tools, designing a microprocessor shouldn't be complicated. The Verilog hardware description language (HDL) is one such tool. This book is a practical guide to processor design in the real world. It presents the Verilog HDL in an easily digestible fashion and serves as a thorough introduction about reducing a computer architecture and instruction set to practice. You're led through the microprocessor design process from the start to finish, and essential topics ranging from writing in Verilog to debugging and testing are laid bare.

340 pages • ISBN 978-0-9630133-5-4

£27.90 • US $45.00

Dual-layer DVD: 165 mins. video

18 DVD Modern Valve Electronics

This filmed seminar (presented by Menno van der Veen) starts with a short discussion of the classic approach using valve load line graphs, followed by current sources and current foldback techniques. Next, the negative effect of cathode electrolytics is covered as well as reducing supply voltage interference. With the help of state of the art measurement techniques the (in)correctness of feedback is proven, while also clarifying what's happening deep within the core of the output transformer.

ISBN 978-1-907920-10-3 • £24.90 • US $40.20

19 Improved Radiation Meter

This device can be used with different sensors to measure gamma and alpha radiation. It is particularly suitable for long-term measurements and for examining weakly radio-active samples. The photodiode has a smaller sensitive area than a Geiger-Müller tube and so has a lower background count rate, which in turn means that the radiation from a small sample is easier to detect against the background. A further advantage of a semiconductor sensor is that is offers the possibility of measuring the energy of each particle.

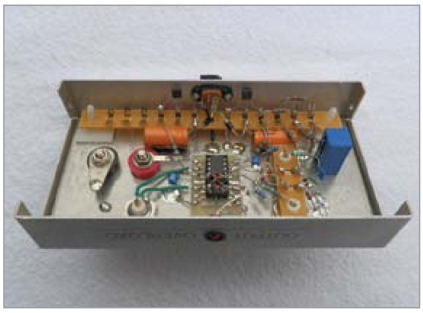
Kit of parts incl. display and programmed controller

Art.# 110538-71 • £35.50 • $57.30

More information on the Elektor Website:
www.elektor.com/store

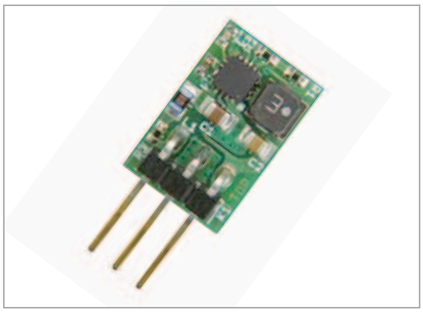
Elektor
Regus Brentford - 1000 Great West Road
Brentford - TW8 9HH - United Kingdom
Tel.: +44 20 8261 4509
Fax: +44 20 8261 4447
Email: order@elektor.com

elektor



Transconductance Amplifier

Although Analog Devices' type AD844 opamp is primarily designed for current-feedback applications, its special structure makes it also very suitable for alternative possibilities in the field of controlled sources. For example, it allows a voltage controlled current source to be designed with ease, for use in measurement technology. Next month we will describe the operation of the AD844 and the construction of an instrumentation amplifier around this opamp.



Cool! 7805 Drop-in

The 7805 is unquestionably the most widely used fixed voltage regulator. Unfortunately this golden oldie has a habit of turning all the excess energy into heat. Wouldn't it be nice to have a switching power supply controller as a drop-in replacement for the 7805? The switched device is sure to offer much higher efficiency as well as better specifications. In the Elektor labs, a small printed circuit board was designed around a TPS62150 buck converter capable of turning an input voltage of 5.5 to 17 V into a 5 V stabilised rail at up to 1 A. By adjusting a number of resistance values other output voltages are also possible.

Extensions for Elektor Improved Radiation Meter

The radiation meter from Elektor November 2011 is used by many readers. The circuit is especially useful in studies of weak radioactive substances and measurements over time. The microcontroller in this circuit has a bootloader, so the program can easily be tweaked. Next month we will discuss some possible adaptations. In addition, we describe some hardware changes to enable the measuring range to be extended.

Article titles and magazine contents subject to change; please check the Magazine tab on www.elektor.com

Elektor UK/European October 2012 edition on sale September 20, 2012. *Elektor USA October 2012 edition published September 17, 2012.*

w.elektor.com www.elektor.com www.elektor.com www.elektor.com www.elektor.com w

Elektor on the web

All magazine articles back to volume 2000 are available individually in pdf format against e-credits. Article summaries and component lists (if applicable) can be instantly viewed to help you positively identify an article. Article related items and resources are also shown, including software downloads, hyperlinks, circuit boards, programmed ICs and corrections and updates if applicable.

In the Elektor Shop you'll find all other products sold by the publishers, like CD-ROMs, DVDs, kits, modules, equipment, tools and books. A powerful search function allows you to search for items and references across the entire website.

Also on the Elektor website:

- Electronics news and Elektor announcements
- Readers Forum
- PCB, software and e-magazine downloads
- Time limited offers
- FAQ, Author Guidelines and Contact

ORDERING INSTRUCTIONS, P&P CHARGES

Non-online orders, except for memberships (for which see below), must be sent BY POST or FAX to Elektor International Media, 78 York Street, London W1H 1DP, United Kingdom.
Online ordering: www.elektor.com/store
Orders placed on our London office must include P&P charges (Priority or Standard) as follows:
Europe: £6.00 (Standard) or £7.00 (Priority). Outside Europe: £9.00 (Standard) or £11.00 (Priority)

HOW TO PAY

All orders must be accompanied by the full payment, including postage and packing charges as stated above or advised by Customer Services staff.
Bank transfer into account no. 40270211 held by Elektor International Media BV with The Royal Bank of Scotland, London.
IBAN: GB96 ABNA 4050 3040 2702 11. BIC: ABNAGB2L. Currency: sterling (UKP).
Please ensure your full name and address gets communicated to us.
Cheque sent by post, made payable to Elektor Electronics. We can only accept sterling cheques and bank drafts from UK-resident customers or members. We regret that no cheques can be accepted from customers or members in any other country.
Credit card VISA and MasterCard can be processed by mail, email, web, fax and telephone. Online ordering through our website is SSL-protected for your security.

COMPONENTS

If difficulties are envisaged in the supply of components for Elektor projects, a source will normally be advised in the article.

Note, however, that the source(s) given is (are) not exclusive.

TERMS OF BUSINESS

Delivery Although every effort will be made to dispatch your order within 2-3 weeks from receipt of your instructions, we can not guarantee this time scale for all orders.

Returns Faulty goods or goods sent in error may be returned for replacement or refund, but not before obtaining our consent. All goods returned should be packed securely in a padded bag or box, enclosing a covering letter stating the dispatch note number. If the goods are returned because of a mistake on our part, we will refund the return postage.

Damaged goods Claims for damaged goods must be received at our London office within 10 days (UK); 14 days (Europe) or 21 days (all other countries).

Cancelled orders All cancelled orders will be subject to a 10% handling charge with a minimum charge of £5.00.

Patents Patent protection may exist in respect of circuits, devices, components, and so on, described in our books and magazines. Elektor does not accept responsibility or liability for failing to identify such patent or other protection.

Copyright All drawings, photographs, articles, printed circuit boards, programmed integrated circuits, diskettes and software carriers published in our books and magazines (other than in third-party adverti-sements) are copyright and may not be reproduced or transmitted in any form or by any means, including photocopying and recording, in whole or in part, without the prior permission of Elektor in writing. Such written permission must also be obtained before any part of these publications is stored in a retrieval system of any nature. Notwithstanding the above, printed circuit boards may be produced for private and personal use without prior permission.

Limitation of liability Elektor shall not be liable in contract, tort, or otherwise, for any loss or damage suffered by the purchaser whatsoever or howsoever arising out of, or in connexion with, the supply of goods or services by Elektor other than to supply goods as described or, at the option of Elektor, to refund the purchaser any money paid in respect of the goods.

Law Any question relating to the supply of goods and services by Elektor shall be determined in all respects by the laws of England.

August 2012

MEMBERSHIP RATES FOR ANNUAL MEMBERSHIP		
	Standard	**Plus**
United Kingdom & Ireland	£54.00	£66.50
Surface Mail Rest of the World	£68.50	£81.00
Airmail Rest of the World	£86.00	£98.50
USA & Canada	See www.elektor.com/usa for special offers	

HOW TO PAY

Bank transfer into account no. 40270211 held by Elektor International Media BV with The Royal Bank of Scotland, London.
IBAN: GB96 ABNA 4050 3040 2702 11. BIC: ABNAGB2L.
Currency: sterling (UKP). Please ensure your full name and address gets communicated to us.

Cheque sent by post, made payable to Elektor Electronics. We can only accept sterling cheques and bank drafts from UK-resident customers or members. We regret that no cheques can be accepted from customers or members in any other country.

Credit card VISA and MasterCard can be processed by mail, email, web, fax and telephone. Online ordering through www.elektor.com/store is SSL-protected for your security.

MEMBERSHIP CONDITIONS

The standard membership order period is twelve months. If a permanent change of address during the membership period means that copies have to be despatched by a more expensive service, no extra charge will be made. Conversely, no refund will be made, nor expiry date extended, if a change of address allows the use of a cheaper service.

Student applications, which qualify for a 20% (twenty per cent) reduction in current rates, must be supported by evidence of studentship signed by the head of the college, school or university faculty. A standard Student Membership costs £43.20, a Student Plus Membership costs £55.70 (UK only).

Please note that new memberships take about four weeks from receipt of order to become effective.

Cancelled memberships will be subject to a charge of 25% (twenty-five per cent) of the full membership price or £7.50, whichever is the higher, plus the cost of any issues already dispatched.

Memberships cannot be cancelled after they have run for six months or more.